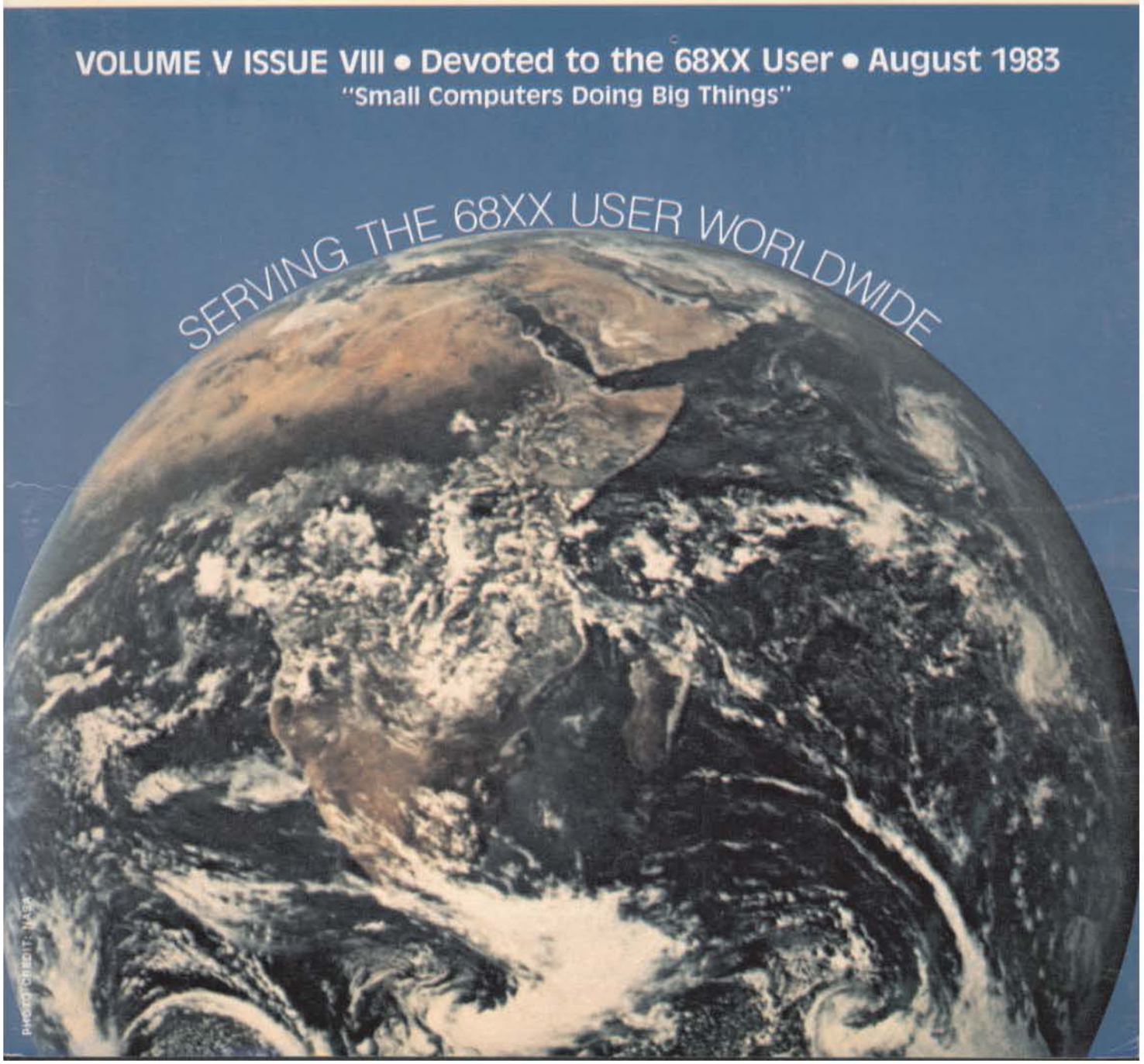
'68' MICRO JOURNAL

$2.95 USA

Australia	A $4.75	New Zealand	NZ $ 6.50
Singapore	S $9.45	Hong Kong	H $23.50
Malaysia	M $9.45	Sweden	30:-SEK

VOLUME V ISSUE VIII • Devoted to the 68XX User • August 1983

"Small Computers Doing Big Things"



YOUR CHOICE-smart either way

- Over 140 software driven functions
- 82 x 24 or 82 x 20 screen format — software selectable
- High resolution 7 x 12 matrix characters — P-31 green phosphor
- Upper/lower case character set — plus graphics character set
- 56-key alphanumeric keyboard — plus 12-key cursor, numeric pad
- Internal editing functions — insert, delete, scroll, roll, slide, etc.
- Parallel printer I/O port
- 50 to 38,400 baud operation — programmable
- Cursor type, cursor position, print control characters, protected fields, shift inversion, dual intensity and many other features

8212 — twelve-inch diagnonal screen or **8209** — nine-inch diagnonal screen

SWTPC

SOUTHWEST TECHNICAL PRODUCTS CORPORATION
219 W. RHAPSODY
SAN ANTONIO, TEXAS 78216 (512) 344-0241

The UniFLEX™ Operating System extracts every last drop

from the 8 bit 6809 microprocessor allowing it to outperform many 16 bit systems

With the UniFLEX™ Operating System, the 8 bit 6809 microprocessor can perform as well as larger CPUs in a multi-user, multi-tasking environment.

Independently developed from the ground up, UniFLEX™ closely models the features found in the UNIX™ Operating System. And in two years of use, UniFLEX™ has proven the abilities of the 6809 to perform large system functions when incorporated into a properly designed mainframe.

Some of the features supported include:

- full multi-user, multi-tasking capabilities
- hierarchical file systems
- device independent I/O
- four Gigabyte disk capacities
- full file protection
- inter-task communication via pipes
- I/O redirection
- task swapping for efficient memory usage
- full random-access files
- comprehensive shell command language
- foreground-background jobs
- electronic mail and printer spooling
- system accounting facilities

The support software currently available for use under UniFLEX™ is extensive. A sampling of the programs available includes:

- native C compiler (full implementation)
- native Pascal compiler
- FORTRAN 77 ANSI Subset compiler
- COBOL compiler with ISAM files, Report Writer & Sort/Merge
- Extended BASIC interpreter
- Extended BASIC precompiler
- text editing and processing software
- enhanced printer spooler
- variety of absolute and relocatable assemblers
- debug and diagnostic packages

Technical Systems Consultants, inc. also offers a line of single user FLEX™ software products for 6800 and 6809 processors. For those having an absolute need for a 16 bit processor, UniFLEX™ will be available through OEM licensing arrangements for the 68000 microprocessor. Please call or write for additional information on individual products or OEM licensing arrangements.

UNIX™ is a trademark of Bell Laboratories.
FLEX™ and UniFLEX™ are trademarks of Technical Systems Consultants, inc.

technical systems consultants, inc.

111 Providence Road
Chapel Hill, North Carolina 27514
(919) 493-1451

'68'

MICRO JOURNAL

Portions of the text for 68 MICRO JOURNAL was prepared using the following furnished hard/software.

COMPUTERS-HARDWARE
Southwest Technical Products
219 W. Rhapsody
San Antonio, Texas 78216
S09-5/8 DMF disk-CDS1-8212W-Sprint 3 Printer

GIMIX Inc.
1337 West 37th Place
Chicago, IL 60609
Super Mainframe-OS9-FLEX-Assorted Hardware

EDITORS-WORD PROCESSORS
Technical Systems Consultants, Inc.
111 Providence Road
Chapel Hill, NC 27514
FLEX-Editor-Processor

Great Plains Computer Company, Inc.
PO Box 916
Idaho Falls, ID 83401
STYLO-Mail Merge

CONTENTS

Send All Correspondence To:

Computer Publishing Center
68 MICRO JOURNAL
5900 Cassandra Smith
PO Box 849
Hixson, TN 37343
615 842-4600

68' Micro Journal is published 12 times a year by Computer Publishing Inc. Second Class Postage Paid ISSN 0194-5025 at Hixson, Tenn. and additional entries. Postmaster: send Form 3579 to 68' Micro Journal, PO Box 849, Hixson, Tennessee.

SUBSCRIPTION RATES

USA

1-Year $24.50 2-Years $42.50 3-Years $64.50

FOREIGN

See Page 52

Items Submitted for Publication

Articles submitted for publication should be accompanied by the authors **full name, address, date and telephone number.** It is preferred that articles be submitted on either 5 or 8 inch diskette in TSC Editor format or STYLO format. All diskettes will be returned.

The following TSC Text Processor commands ONLY should be used (due to our proportional processor): .sp space, .pp paragraph, .fi fill and .nf no fill. Also please do not format within the text with multiple spaces. The rest we will enter at time of editing.

STYLO commands are all acceptable except the .pg page command, we print edited text files in continous text.

All articles submitted on diskettes should be in TSC FLEX™ format, either FLEX2 6800, or FLEX9 6809 any version.

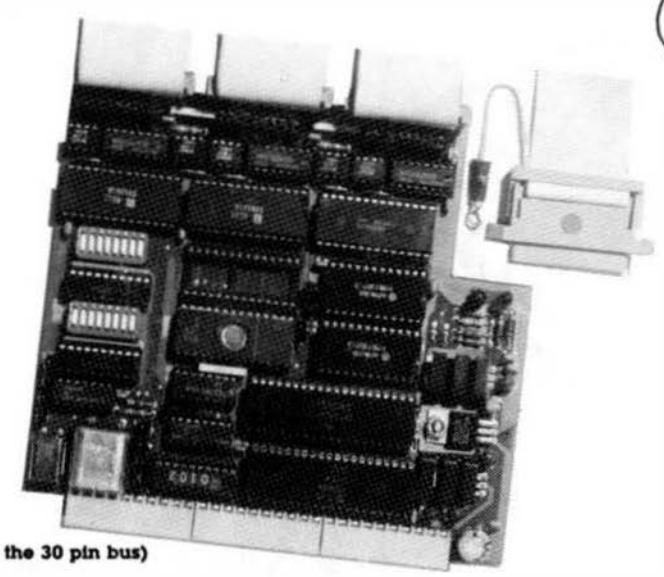
If articles are submitted on paper they should be on white 8X11 bond or better grade paper. No hand written articles (hand written or drawn art accepted). All paper submittad articles will be photo reproduced. This requires that they be typed or produced with a dark ribbon (no blue), single spaced and type font no smaller than 'elite' or 12 pitch. Typed text should be approximately 7 inches wide (will be reduced to column width of 3 1/2 inches). **Please use a dark ribbon!**

All letters to the editor should also comply with the above and bear a signature. Letters of 'gripes' as well as 'praise' are solicited. We attempt to publish all letters to the editor verbatim, however, we reserve the right to reject any submission for lack of 'good taste'. We reserve the right to define what constitutes 'good taste'.

Advertising: Commercial advertisers please contact the 68 Micro Journal advertising department for current rate sheet and requirements.

Classified: All classified must be non-commercial. Maximum 20 words per classified ad. Those consisting of more than 20 words should be figured at .35 cents per word. 20 words or less $7.50 minimum, one time, paid in advance. No classified ads accepted by telephone.

Intelligent Serial I/O Processor Board #11



(for the 30 pin bus)

The GIMIX Intelligent Three-port RS-232C Serial Interface can significantly increase the throughput of a multi-user system. By buffering data transfers between users and the system, and preprocessing the data, it reduces the number of interrupts to the host CPU, allowing the host more time for other processing tasks.

Features:

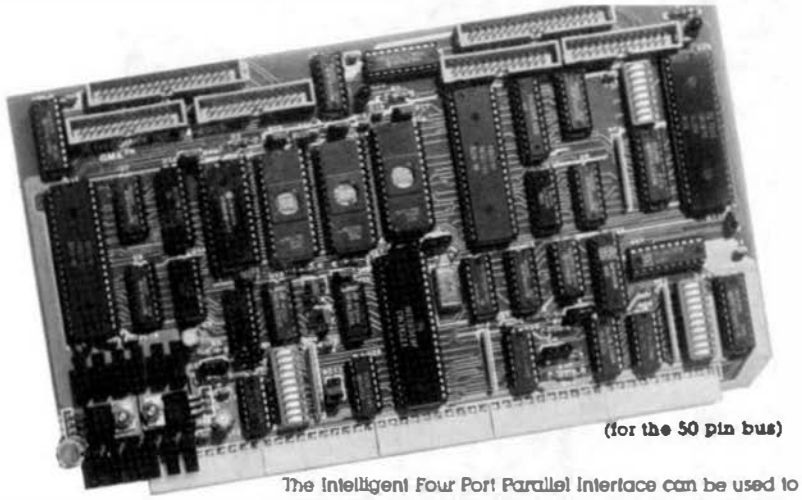
- Independent on-board 2MHz 68809 CPU
- Up to 20K of on-board memory (EPROM and RAM)
- Buffered data transfer between host and on-board CPUs using a 28038 FIO with 128 byte bi-directional FIFO buffer and mailbox message capabilities
- Three RS-232C serial I/O ports (6551As) with software selectable baud rates, word length, stop bits, and parity
- Each port has five "handshake" lines for modem control applications
- The on-board 6809 can be reset by the host processor
- Compatible with memory-to-memory DMA transfers to/from the GMX 6809 CPU III
- Sense switches and status LEDs that can be used to select software options and indicate board status
- Separate 26-pin cable connections for each port

Appropriate on-board firmware and operating system drivers are required. Uses up to three #95 cable sets (DB-25S connectors).

OS-9 firmware and drivers for the Intelligent 3-port Serial Interface

The OS-9 firmware and drivers enhance the performance of multi-user systems, while providing functions equivalent to the standard ACIA type drivers normally used for terminals and serial printers. Input line editing functions (backspace, echo, line dup and repeat, etc.) are handled by the I/O board, rather than the host, allowing the host more time for other processing tasks. The host is only interrupted when a complete input line (terminated by a "CR") is entered, or certain special characters are received. Input and output data are buffered on the I/O board so that the host can perform other tasks while serial data is being transmitted or received. When used with the GMX CPU III, block data transfers between the I/O board and the host use the CPUs memory-to-memory DMA to further enhance throughput. In addition to performance enhancements, features such as software selectable baud rates and transmission characteristics (number of data bits, stop bits, parity, etc.) are provided. The board also transmits "messages" to any or all I/O ports to indicate that the I/O interface is ready and "waiting for the host", and that the host is "on-line" and has opened a path to the port. Messages to individual ports can be disabled.

Intelligent Parallel I/O Processor Board #12



(for the 50 pin bus)

The Intelligent Four Port Parallel Interface can be used to improve system performance by buffering data transfers to parallel peripherals such as printers and/or by buffering and pre-processing parallel input data from keyboards, sensors, etc.

Features:

- Independent on-board 2MHz 68809 CPU
- Up to 32K of on-board memory (EPROM and RAM)
- Buffered data transfer between host and on-board CPUs using a 28038 FIO with 128 byte bi-directional FIFO buffer and mailbox message capabilities.
- Four fully buffered 8-bit parallel ports with handshaking and input/output latches (two 6522 VIAs). Each 6522 also has two 16-bit counter/timers and a shift register for serial data transfers.
- Software programmable direction for each bit on two of the four ports (1 per VIA), the other 2 ports can be individually programmed as 8 in or 8 out. The bi-directional handshake lines can be programmed as inputs or outputs.
- The on-board 6809 can be reset by the host processor.
- Full 20-bit address decoding, it can be addressed on any 4 byte boundary in 1M byte of address space.
- Compatible with memory-to-memory DMA transfers to/from the GMX 6809 CPU III.

Appropriate on-board firmware and operating system drivers are required. Uses up to four #95 cable sets (DB-25P connectors) or two 36-pin cable sets with Centronics compatible connectors. Centronics compatible cable sets include a back panel connector plate for the Classy Chassis. Back panel to printer cables are also available.

GIMIX INC 1337 West 37th Place • Chicago, Illinois 60609 • (312) 927-5510 • TWX 910-221-4055

FLEX™ USER NOTES
THE 6800-6809 BOOK

By: Ronald W. Anderson
As published in 68 MICRO JOURNAL™

The publishers of 68 MICRO JOURNAL are proud to announce the publication of Ron Anderson's **FLEX USER NOTES,** in book form. This popular monthly column has been a regular feature in 68 MICRO JOURNAL SINCE 1979. It has earned the respect of thousands of 68 MICRO JOURNAL readers over the years. In fact, Ron's column has been described as the 'Bible' for 68XX users, by some of the world's leading microprocessor professionals. Now all his columns are being published, in whole, as the most needed and popular 68XX book available. Over the years Ron's column has been one of the most popular in 68 MICRO JOURNAL. And of course 68 MICRO JOURNAL is the most popular 68XX magazine published.

As a SPECIAL BONUS all the source listing in the book will be available on disk for the low price of: FLEX™ format only — 5" $12.95 — 8" $16.95 plus $2.50 shipping and handling, if ordered with the book. If ordered separately the price of the disks will be: 5" $17.95 — 8" $19.95 plus $2.50 shipping and handling.

Listed below are a few of the **TEXT** files included in the book and on diskette.

All TEXT files in the book are on the disks.

LOGO.C1	File load program to offset memory — ASM PIC
MEMOVE.C1	Memory move program — ASM PIC
DUMP.C1	Printer dump program — uses LOGO — ASM PIC
SUBTEST.C1	Simulation of 6800 code to 6809, show differences — ASM
TERMEM.C2	Modem input to disk (or other port input to disk) — ASM
M.C2	Output a file to modem (or another port) — ASM
PRINT.C3	Parallel (enhanced) printer driver — ASM
MODEM.C2	TTL output to CRT and modem (or other port) — ASM
SCIPKG.C1	Scientific math routines — PASCAL
U.C4	Mini-monitor, disk resident, many useful functions — ASM
PRINT.C4	Parallel printer driver, without PFLAG — ASM
SET.C5	Set printer modes — ASM
SETBAS1.C5	Set printer modes — A-BASIC
	(And many more)

**Over 30 TEXT files included in ASM (assembler) — PASCAL — PIC (position independent code) TSC BASIC-C, etc.

NOTE: .C1,.C2, etc. = Chapter 1, Chapter 2, etc.

This will be a limited run and we cannot guarantee that supplies will last long. Order now for early delivery.

Foreign Orders Add $4.50 S/H

Softcover — Large Format

Book only: **$7.95** + $2.50 S/H

With disk: 5" **$20.90** + $2.50 S/H

With disk: 8" **$22.90** + $2.50 S/H

See your local S50 dealer/bookstore or order direct from:

Computer Publishing Inc.
5900 Cassandra Smith Rd.
Hixson, TN 37343
(615) 842-4601

'*FLEX is a trademark of Technical Systems Consultants

Universal Data Research, Inc. Introduces

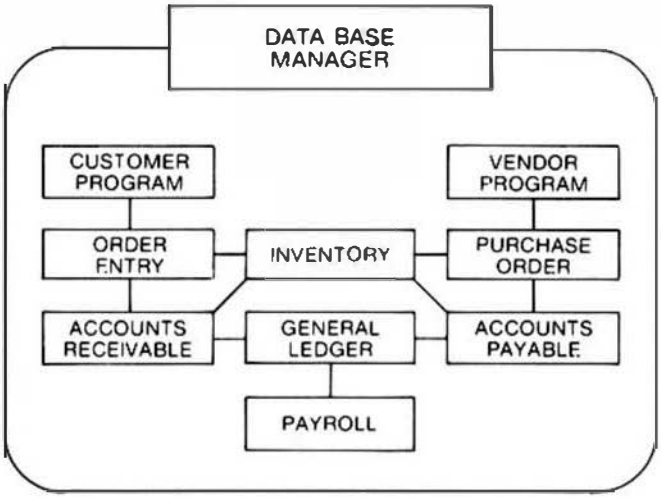
MODEMS

300 Baud Acoustic	$149.00
300 Baud Direct	$179.00
300 Baud Auto Answer	$219.00
1200 Baud Direct	$449.00
1200 Baud Auto Answer	$499.00
300/1200 Auto Answer	$549.00
300/1200 Auto Dial	$599.00
Modem 2 Port I/O Card	$119.00
with speed select	
Printer Special: Okidata 82A	$449

Dealer Inquiries Welcome ● Call or Write For Free Catalogue

New, Lower Prices

FLEX* and UniFLEX*
software for the 6809



Integrated Business Programs

	FLEX	UniFLEX
Accounts Payable	$295	$395
Accounts Receivable	$295	$395
General Ledger	$295	$395
Inventory 2	$295	$395
Payroll	$295	$395
Data Base Manager	$350	$450
Word Processing		
Software	$295	$395
WP Menu		$150
P Control		$150

UNIVERSAL DATA RESEARCH INC.

*FLEX & UniFLEX are Trademarks of Technical Systems

2457 Wehrle Drive, D-68, Buffalo, NY 14221
PHONE (716) 631-3011

Dealer Inquires Welcome

master charge VISA

TMP (Total Management Planning) OS-9 SOFTWARE

THESE OFF-THE-SHELF APPLICATION PACKAGES HAVE CHANGED THE PICTURE FOR PROFESSIONAL SOFTWARE USERS

Here are the comprehensively supported OS-9 application software packages that discerning professionals are selecting today. Each one runs stand-alone or in combination with other TMP packages.

TMP/CALC is a *new-generation* "Electronic Spreadsheet" package that goes a quantum leap beyond ordinary Calc software. Superior speed, and extra features like Dynamic Calculations and Dynamic Overlay, mean you can change a cell *anywhere* on the spreadsheet and get automatic updates wherever they belong. No more "wrap-around" since output to the printer is formatted. "Help" screens prompt without erasing data, and the rows and columns on any worksheet are limited *only* by available memory.
ONLY $250

TMP/Manager is a structured database manager, and is the ultimate driving force behind any total *Management/Marketing Planning* system. Its built-in select/sort/merge module — and optimized machine code — makes it superfast, and provides unusually nimble data manipulation characteristics. Other outstanding features are detailed prompting, easy identification of data fields, and an excellent security system.
ONLY $300

TMP FreeForm fills the void between a structured DBMS and word-processing; it's an "electronic index card" package with an endless list of valuable applications such as parts cataloging, look-up inventories and listings of *any* kind. Type anything on up to 32,000 electronic "cards" of up to 9 pages each that reside in "file drawers" (databases). Up to 13 key-words can call each page.
ONLY $150

TMP/Front-End integrates any or all of the packages described above, and integrates TMP with word-processors, BASIC and other high-level languages.
NO CHARGE with TMP/Manager

TMP T.M.

TMP PACKAGES ARE EXCEPTIONALLY FRIENDLY. EACH COMES WITH MULTI-MEDIA TRAINING AND OUTSTANDING DOCUMENTATION

AUDIO CASSETTE TRAINING TAPES are provided with each TMP software package. These crisp, professionally produced training aids take a new user through "hands-on" exercises that instantly build confidence, and minimize reference to manuals.

CLEAR DOCUMENTATION is specially prepared for each TMP package. Remarkably readable because they are conversational in style and logically organized, this series of manuals *further* assures quick user comfort and productivity with TMP software. Includes sections on how to best utilize the system; integrating user-written software with TMP packages, and other bonus topics.

VIDEOTAPE PROGRAMS are available to TMP Dealers. They dramatically take the viewer into real business environments for a close look at how a TMP software package is utilized. Video programs can be personalized for specific organizations. Programs may be specified for most popular tape formats.

THE BOTTOM-LINE IS THIS: Software is only as effective as its training and documentation. TMP packages are *incomparable* in both areas.

RUN TMP IN A SMOKE SIGNAL CHIEFTAIN™ COMPUTER TO EXPERIENCE THE KIND OF SPEED AND POWER YOU CAN EASILY AFFORD *TODAY*

Endurance-Certified Chieftains are consistently among the fastest computers in CPU and I/O speeds according to the widely respected BENCHMARK REPORTS. That, combined with almost legendary reliability, makes Chieftains the logical choice in computers using OS-9 operating systems.

Configurations range from floppy-disk systems to multi-user, multi-tasking Winchester hard disk systems with tape back-up, performance approaching mainframes at prices you can live with.

TMP software packages for most OS-9 systems and Chieftain computers are offered exclusively from Smoke Signal. Inquiries from non-Smoke Signal dealers are invited. Call (213) 889-9340.

SMOKE SIGNAL
Chieftain Computers
31336 Via Colinas • Westlake Village, CA 91362
(213) 889-9340

ELEKTRA Means Quality

ELEKTRA CABINET Made of heavyweight 0.090" thick aluminum. Interior is 18-1/2" wide by 21-7/8" deep by 6-3/4" high. Heavy duty A.C. line cord. A.C. fuse holder. EMI filter. Fan with filter. Back panel has 10 cutouts for 'D' type data connectors. Front panel has key on/off power switch, 2 illuminated push button switches (Reset and NMI/Abort), and two cutouts for 5-1/4" disk drives. $250.00

Filler Plate for 5-1/4" drive opening: $10.00 Fan Filter: $10.00

POWER SUPPLY Highest quality linear power supply CONSERVATIVELY rated at 15a @ 8v, 3a @ 16v, 3a @ -16v. 3 primary inputs for light, rated, and heavy loading.
220v Version: $200.00 110v Version: $175.00

DISK REGULATOR BOARD WITH CABLES Standard version for 2 floppy drive: $50.00
Heavy duty version for 1 Winchester drive and 1 floppy drive: $75.00

ELEKTRA UNIVERSAL MOTHERBOARD Heavyweight 0.125" thick, 18" long by 8" wide, 11 memory (50 pin) slots. 4 or 8 slots may be cut off for shortening to 14" or 10" lengths respectively. 8 I/O (30 pin) slots. Complete address decoding of I/O slots. Choice of 4, 8, or 16 addresses per I/O slot. Base address for I/O slots can be placed at 32, 64, or 128 byte increments respectively. 1" spacing between all memory and I/O slots. Extended addressing capability for both memory and I/O ports for meeting SS-50C bus specifications. On board baud rate generator with low and high ranges, providing jumper selectable rates of 75 through 38,400 for each of the five baud rate lines. Slow device circuitry permitting 1MHz 30 pin disk controllers to run with 2 MHz 50 pin CPU boards.
Bareboard w/ documentation: $80.00
Kit w/gold connectors: $320.00 Assembled w/gold connectors: $380.00
Kit w/tin connectors: $24 .00 Assembled w/tin connectors: $300.00

ELEKTRA CHASSIS Includes cabinet, 110v power supply, power supply cables, standard disk regulator board with power cables, motherboard with gold square pin connectors, assembled and tested. $850.00

ELECTRA CPU 6/9 Use either the 6802 or 6809 (to run 6800 software) or 6809. Has provision for up to 3 27 16 Eproms, 1K scratchpad, M 6840 triple timer, and an optional baud rate generator providing baud rates from 110 through 38,400 baud in two user selectable ranges. The board supports DMA by either HALT or BUSREQ when a 6809 CPU is used. The board will run any of the MIKBUG™ compatible monitors in the 6802/6808 mode and SBUG-E, HUMBUG, and MICROBUG in the 6809 mode. Versions of OS-9™ Level 1 are available.
Bar board: $90.00 Kit: $225.00 Assembled: $275.00
2K 6800 HUMBUG: $40.00 2K 6809 MICROBUG: $30.00 See 4K HUMBUGS next page.
Optional Baud Rate Generator: $25.00

ELEKTRA DPS DUAL PORT SERIAL CARD Fits the standard 30 pin SS-50 bus I/O slot. Can be configured for 4 addresses per port with the B port 2 addresses higher than the A port or for 16 addresses per port with the B port 4 addresses higher than the A port. Each port is terminated at two 16 pin dip sockets, one socket configured for modem and the other for terminal or printer. RTS, CTS, DTR, DCD, and DTR are appropriately implemented. Each port has independent selection of baud rate. Each port allows the interrupt request to be independently jumpered to the IRQ or FIRQ/NMI bus line.
Bareboard: $20.00 Kit: $60.00 Assembled: $80.00
Cables (two needed per board): Each: $20.00

ELEKTRA DPP DUAL PORT PARALLEL CARD Fits the standard 30 pin SS-50 bus I/O slot. Can be used in either the 4 or 16 addresses per I/O slot configuration occupying the first four addresses of the I/O slot. The direction of the TTL buffers can be controlled by either on board jumper connections or by a signal from the peripheral. The interrupt request lines for each port may be individually jumpered to either the IRQ or FIRQ/NMI bus line.
Bareboard: $20.00 Kit: $60.00 Assembled: $80.00
Cables (two needed per board): Each: $20.00

ELEKTRA D-5 Dual drive cabinet for 5-1/4" drives with power supply, line cord, fuse, power switch, and power cable to drives: 125.00
ELEKTRA HD-5 (Heavy duty version of D-5 package above): 150.00
ELEKTRA SHD-5 (Super heavy duty. Powers 1 Winchester and 1 floppy): 175.00
5' ribbon cable for dual 5 1/4" disk drives: 40.00
ELEKTRA D-8 Dual drive cabinet, power supply, pa cable for 8" drives: 350.00
6' ribbon cable for dual 8" disk drives: 45.00

GOLD 10 PIN CONNECTORS (Specify male w/square pins or female): 1.50
TIN 10 PIN CONNECTORS (Specify male w/square pins or female): 0.50

Prices and inventory are subject to change without advance notice.
This ad is our catalog.

WARNING AAA Chicago Computer Center does not provide repair or diagnostic service for customer assembled kits. AAA Chicago Computer Center does warranty and maintain service for our assembled boards. The customer should carefully take into consideration the small differential separating our kit and assembled prices when making his choice of purchase.

We have introduced our line of computer equipment with the purpose of offering the highest quality of components possible at affordable prices. These products are intended for OEM applications where it is the responsibility of the purchaser to integrate these components with suitable memory, disk controllers, drives, and software along with I/O terminals to form working computer systems.

AAA CHICAGO COMPUTER CENTER
120 CHESTNUT LANE • WHEELING, IL 60090
(312) 459-0450
Technical consultation availabl 4 PM to 6 PM most weekdays. Closed ev ings and weekends.

TERMS Minimum order $20.00 Shipping and handling estimates within the Continental U.S., add 3% (MINIMUM $2.50). Illinois residents add 6% sales tax. We will refund your overestimated shipping and handling charges. Foreign shipping and handling, add 10% (MINIMUM $10.00). Foreign orders must be prepaid in U.S. dollars. Heavy foreign items will be shipped air freight collect. Please phone between 4 PM and 6 PM weekdays if questions arise regarding shipping fees. Master Charge, Visa, and American Express honored.

Our apology. We are not staffed to answer technical inquiries through the ...il. Please phone for technical help during the hours indicated above. The too frequent changing of our inventory and prices makes it uneconomical to publish a catalog. Our ads are intended to serve that purpose. Prices and inventory are subject to change without advance notice.

SD/DD DISK DRIVES	1 head CDC	2 heads CDC	1 head MPI	floppy Siemens	2 heads MPI	2 heads QUME
30 day guarantee						
5-1/4", 40 tracks	225.00	299.00	250.00	260.00	325.00	350.00
5-1/4", 80 tracks	N/A	375.00	325.00	335.00	425.00	450.00

MPI Service Manual (Specify 40 or 80 track): 20.00
8", 77 tracks DS/DD: Qume DT-8 $525.00 Remex (Special) 350.00

ELEKTRA
COMPUTER PRODUCTS

SMOOTH™ Software

(All of our software is copyrighted and all rights are reserved. Source is either supplied or optionally available at extra cost so that the purchaser can modify our programs for his own use. Licensing, however, is required for commercial resale.)

SUPER MODEM PROGRAM

Single character commands for simplicity.

Hardware independent (No interrupts required). Assumes modem is connected to an MC6850 (serial interface) and the control terminal is connected to an MC6850 (or MC6820 when used with the video version of GMXBUG™).

Transmit manually to distant computer.

Transmit disk files (text) of any length to distant computer.

Receive and save disk files (text) of any length on local disk system. If sending computer does not support an X-on/X-off protocol, then the received files are limited in size by the computer memory.

Tested for full duplex operation at speeds up to 9600 baud. (CRT terminal must be capable of operating at a baud rate higher than the one the modem is operated at.)

Half duplex option in case distant computer doesn't echo.

Echo option so user can simulate a time sharing system. (Super Modem program doesn't support auto-answer but the source is provided for those individuals who wish to adapt our program to their special needs.)

Replaces LF with CR/LF (user option) for those using time sharing systems that don't transmit LF's.

Slow disk file transmit based on character verification (plus user installed timing loops if necessary) for use on time sharing systems to which disk files cannot be sent at speeds suggested by the baud rate.

Please specify 6800 or 6809, SSB or FLEX™, 5" or 8".
Manual and disk with th source and object code: $75.00

STANDARD MODEM PROGRAM

Same as Super Modem Program above but without ECHO option, CR/LF for CR option, slow disk file transmit option, nor X-on/X-off option. Reception of disk files is limited to those small enough to completely fit within the receiving buffer.

Please specify 6800 or 6809, SSB or FLEX™, 5" or 8".
Manual with instructions, source listing, and flow chart; disk with both source and object code: $45.00
Manual with instructions, source listing, and flow chart: 25.00

MODEMS (By U.S. Robotics) Auto-Dial is Hayes compatible	MANUAL ANSWER	AUTO ANSWER	AUTO-ANS AUTO-DIAL
300/1200 Baud, direct connect	N/A	449.00	499.00
1200 Baud (120 cps), direct connect	349.00	399.00	N/A
300 Baud (30 cps), direct connect	149.00	179.00	N/A
300 Baud (30 cps), acoustic	119.00	N/A	N/A

We will be closed from July 30th through August 21st.

THESE ADS ARE OUR CATALOG

GIMIX™ CLEARANCE SALE

	LIST	OUR PRICE		LIST	OUR PRICE
Relay driver board	538 86	250 00	6800 CPU board (used)	244 03	125 00
Filler Plate	14 92	10 00	Baud rate gen. board	88 93	65 00
Cable (Ser or Par I/O)	24 95	20 00	#48 floppy controller	198 48	150 00
#58 floppy controller	226 58	175 00	#28 control w/GMX FLEX	328 28	270 00
Double disk reg. card	68 22	50 00	#58DMA con w/GMX FLEX	618 68	470 00
32K (with 16K $125 00)		175 00	56K memory board		425 00
64K memory board	478 87	450 00	Single prt. ser., 1 cable	113 38	90 00
Dual prt. ser., 2 cables	178 33	140 00	Dual prt. par. w/2 cables	138 32	110 00
Mainframe	1398 19	950 00	#29 Computer System	2498 29	2000 00
#59 Computer System	2988 59	2400 00	#49 Computer System	3248 49	2600 00
64 X 16 Video Boards	198 71	100 00	4K PPD PROM Bd. and Burner		125 00
16K Mem Bds., most w/cntrl reg.		145 00	Reset/Abort Toggle Switch		20 00
93L422 RAM chips (2 needed for GIMIX DAT)				each	20 00

SMOOTH™ Software

ALL IN ONE
Editor - Text Processor - Mailing Labels
Mailing Lists - Use any CRT terminal and printer

Supports Editing commands such as bottom, change, delete, find, insert (single line), input (multiple lines), list, next, overlay (with cursor editing, character deletion and insertion), overstrike (for selected darker text), print, restart, set, top, underline, up, and verify.

Supports Text Processing commands such as block copy, block mo e, centering margin justification (widen and narrow), pagi g, and tabbing.

Mailing Lists and Labels. Use the same mailing list disk file (with protected areas) for both mailing labels and repeat letters. Repeat letters are personally adressed to each person or selected persons on the mailing list.

Most Powerful File Handler found in any editor. Append one file to the end of another, or insert (merge) one file into another as designated by the line pointer. Print specified lines to your printer or to a disk file. Edit files larger than the text buffer. Does not produce output files when not desired. Delete disk files from the editor.

Printer commands. C trol character can be sent to the printer for formal control either directly from the control terminal or by imbedding them in the text. The set command contains interface ini alization and character output routines to support the SWTPC MP-C interface as well as the standard serial and parallel int erfaces. Jumps are also provided to user supplied printer routines. User selects the port address (0 thru 7, A or B) thereby eliminating the need for the user to install printer soft are routines. Editor can be initialized for either 4 or 16 addresses per port.

Editor allows exiting to either the monitor or DOS and then reenter (Warm Start) without destroying previously prepared text in the buffer. The Restart command erases contents in the buffer without the user having to reload the Editor.

T e Editor allows the user to toggle between full du lex (no echo) and half duplex (echo) as needed. It responds to commands in both upper and lo er case and can be used to create assembler source code and Basic programs as well as text.

Specify 6800 or 6809, SSB or FLEX™, 5" or 8"	75 00
Printed source listing is available for an additional	35 00
All-In-One, Write'n Spell, and Spell'n Fix package	250 00

Software by Technical Systems Consultants, Inc.	UniFLEX™ w/1 yr mnt.	FLEX™
DOS (includes Editor and ssembler)	550 00	150 00
Editor or Assembler		50 00
68000 Cross Assembler on 6809	300 00	250 00
6809 Cro Assembler on 6800 or 6800 FLEX™ Utilities		100 00
Text Processo or Sort/Merge Package or 6809 FLEX™ Utilities	150 00	75 00
Extended Basic	200 00	100 00
Basic Precompiler (specify standard or extended)	150 00	50 00
Pascal	300 00	200 00
Debug Package or Diagnostic Package	75 00	75 00
6809 Relocating Assembler & Linking Loader	175 00	150 00
Fortran (Requires Relocating Assembler & Linking Loader)	350 00	275 00
Fortran (With Relocating Assembler & Linki g Loader)	450 00	375 00
Cobol	750 00	500 00

SMOOTH and **ELEKTRA** are trademarks of AAA Chicago Computer Center
FLEX and **UniFLEX** are trademarks of Technical Systems Consultants, Inc.

6 NEW ELEKTRA PRODUCTS

ELEKTRA 30 PIN PROTOTYPING BOARD 20.00
ELEKTRA 50 PIN PROTOTYPING BOARD 40.00
ENGINEER'S "FUN BOX" by ELEKTRA Computer cabinet with high quality 10 amp power supply, EMI filter, Fan. Large enough to hold the standard size SS-50 type motherboard. 225.00
64K STATIC MEMORY BOARDS with gold connectors (tin available). Assembled and tested. With 56K RAM installed $299.00 With 64K RAM installed $349.00
12 and 24 MEGABYTE ELEKTRA ADD-ON WINCHESTER SYSTEMS (See below for pricing)
ORDER — WRITE UP COMPUTER PROGRAM Screen oriented write up form with cursor editing, disk save and load, printer command using easily available universal print-out forms. Computer program can be used to write up purchase orders, packing list, account statements, plus more. Program is written in 6809 assembler language and does not require reverse video. Routines are included to support screen wrap around in both directions in case your video terminal does not. Source is provided so the customer can modify the program if he wishes. Comments are provided for adaptation to Lear Siegler and Televideo CRT terminals. The program can be easily adapted to other terminals if HOMEUP, HOMEUP and ERASE, and direct cursor addressing codes are known. CR, LF, and BS are the only other codes used. Available for 6809 FLEX. $100.00

SPECIALS

- Modems (Up to $100.00 off list). See our ad on the previous page.
- SSB BFD Floppy DiskControllers (Version 3) Run FLEX or SSB DOS 150.00
- MICROBUG (2K, 6809, Ba y HUMBUG by Peter Stark) 30 00
- Cat 300 Baud Acoustic Modem (only 1 left) 75.00
- SWTPC 4K Memory $15.00, 8K Memory $40.00, MP-C $10.00, MP-LA $60.00
- SWTPC Chassis, PS, MB 125.00 · SWTPC MP-2A (Only 1 left) 150 00
- High speed tape reader $50 00 · MPI or Siemens 5" drives 125 00
- Remex 8" DS/DD (Special) $350 00 · Ribbons for MX-80 $5 99

4K 6809 HUMBUG	75 00
4K 6800 HUMBUG (RAM needed at $A000 and $D000)	65 00
Other HUMBUG versions including video versions are available	
Spell'n Fix by Peter Stark	178 58
Write'n Spell by Peter Stark	75 11
Dyn mit Disassembler	80 00
SUPER SLEUTH Disassembler Sy tem ($101.00 for OS-9 version)	99 00

Software by Microware Systems Corp.	Run-Time Package	Update	Source	Manual Only	Object w/Man.
OS-9™ Level One Operating System		75 00	400 00	40 00	200 00
OS-9™ Level Two Operating System		75 00	N/A	40 00	500 00
BASIC09™	100 00	75 00	N/A	25 00	200 00
OS-9™ Macro Text Editor			300 00	15 00	125 00
OS-9™ Interactive Assembler			300 00	10 00	125 00
OS-™ Interactive Debugger (Disk version)			100 00	10 00	50 00
CIS Cobol Compiler	400 00	50 00	N/A	80 00	900 00
Pascal Compiler	100 00	100 00	N/A	40 00	400 00
Microware yearly support service ($200 00 for OS-9 Level 2)					75 00

SWTPC

6809 SWTPC FLEX™ Disk and manual (Disk only 15 00)			35 00
DC-4 Disk Controller (SS/DS, SD/DD, 5-1/4")			230 00
S32 Universal Static Memory Board (without memory chips)			124 50
SBUG-E (2716 compatible)	25 00	MP-R 2716 Eprom Programmer	114 50
MP-S2 Dual Port Serial	120 00	MP-L2 Dual Port Parallel	120 00
MP-N Calculator Board (kit)	54 95	MP-N (assem led)	92 00
MP-T Interrupt Timer (1 left)	92 00	MP-09 2MHz 6809 CPU Board	295 00

Smoke Signal Broadcasting

DCB-4A Double Density Controller Board for 5" and 8" with DOS	549 00
SSB DOS (Specify 6800 or 6809, BFD or DCB-4A, 5" or 8")	75 00
SE92/SA92-5 16809 Edit/Assm (r DOS)	69 95
SSB Monitor (Specify 6800/6809, $E000/$F7E0)	75 00
SSB version of FLEX™ (without Editor and Assembler)	150 00
LMB-1A Motherboard	399 00
SCB-69 6809 CPU Board	399 00
PAR-1 Dual Port Parallel Board	89 00
SER-2 Dual Port Serial Board with 2 Cables	129 00
Chief 90 64K Computer System	2195 00

OS-9 and BASIC09 are trademarks of Microware Systems Corp.
GIMIX and GHOST are registered trademarks of GIMIX Inc.

THE BEST 30 PIN FLOPPY DISK CONTROLLER THAT YOU CAN BUY! — THE ELEKTRA SUPER FC

Controls up to four 5 1/4" drives and four 8" drives for a total of eight system drives for both 6800 and 6809 systems. (FLEX software has a built in limit of a system mix of four drives). Improved programmed input/output (PIO) interface that allows single and double density 5 inch operation at 1MHz in both 6800 and 6809 systems. Allows single density 8 inch operation at 1MHz in both 6800 and 6809 systems. Allows both single and double density operation of both 5 inch and 8 inch drives at 2MHz in both 6800 and 6809 systems without slow device circuitry. Hardware and software compatible with existing SWTPC DC-1, DC-2, AND DC-3 type controllers. Controls virtually any floppy disk drive, any mix of drive size, recording density, track density, number of sides, motor control, with or without head load solenoids, including the new 2 Mbyte 5 inch drives. Analog phase locked loop data separators for maximum reliability. Provides separate adjustments for 5 inch and 8 inch drives. Analog write precompensation circuit with separate adjustments for 5 inch and 8 inch drives. Designed to meet the data hold requirements of Western Digital floppy disk controller IC. Schmidt trigger hysteresis input buffers for disk drive signals. Software write protect control. Interrupt/Data Request enable control. 12 option jumper areas to select just the right combination of hardware features to match your drives and software. Provides many separate option selections for 5 inch and 8 inch drive operation. Gold bus and drive cable connectors. Test connector for easy setup and field service. Single 5 volt power supply. 4 or 16 addresses per slot decoding. Ideal replacement for existing controllers to add double den ity data storage and 8 inch drive capability. Disk driver software and disk formatting utilities available to extend the capabilities of your current software. Get more for less with the ELEKTRA Super Floppy Controller.

ELEKTRA Super FC Super Fl ppy Controller (assembled and tested)	$250 00
Disk with drivers and formatting utilities (Specify 6800 or 6809)	$30 00
EPROM with driver overlays	$30 00
TSC General FLEX or FLEX 9 both with Editor and Assembler	$150 00

THE BEST WINCHESTER DISK SYSTEM THAT YOU CAN BUY! — THE ELEKTRA WINCHESTER

Has automatic error detection and **CORRECTION** of up to 11 bit burst errors.

SS-50 bus, extended addressing capabilities, DMA, on board sector buffer. Operates with 6800 and 6809 type processors, drivers included for FLEX 9 or OS-9. Drivers for FLEX 2 (6800) available for an additional $100.00. Price includes host interface, controller, drive(s), and cables.

12 Megabyte ELEKTRA single drive Winchester system	$2295.00
24 Megabyte ELEKTRA dual drive Winchester system	$3595.00
19 Megabyte ELEKTRA single drive Winchester system	$2995.00
38 Megabyte ELEKTRA dual drive Winchester system	[illegible]
ELEKTRA heavy duty regulator bd. (powers 1 floppy and 1 Winchester)	$75 00
ELEKTRA SHD-5 cabinet, ps, ps cables (powers 1 floppy and 1 Winchester)	$175.00

AAA Chicago Computer Center See our ad on the previous page to your left for ordering instructions.

Technical Consultation available 4 p.m. to 6 p.m. most weekdays. Closed evenings and weekends.

Flex User Notes

Ronald w. Anderson
3540 Sturbridge Court
Ann Arbor, MI 48105

DISK COMPATIBILITY

A few months ago, I confused everyone with a rather incomplete discussion of disk compatibility between or among FLEX systems. Perhaps this will add to the confusion, but hopefully, it will clarify just a bit. Carl Kreider wrote me recently, and I will quote his explanation of part of the problem.

"The only difficulty is largely out of the users control. The 1771 (DC1, DC2, DC3, DMAF1, PTA, FD1 etc.) accepts either $FF or $00 in certain inter-record gap bytes, while the 179X family (DC4, DMAF2, etc.) require $FF. Since it is easy to clear the memory used for the track buffer during format, early formatters used all $00. This means that most disks formatted on the old hardware, using the 1771, can not be read by a controller using the 179X."

Carl goes on to say that he has circumvented the problem by writing his own formatting software. That explains one of the problems with interchange. The other one seems to me to be a difference in how SWTPC and GIMIX handle directory entries when the directory sectors allocated on track 0 are used up. Generally, disks with less than 60 directo y entries are compatible between the two FLEX systems. Either one will choke on the othe at directory entry 61, and report an error. I have reason to believe that the SWTPC method is to go grab the next free sector and link the last filled directory sector to it. What I am describing may only be applicable to double sided disks, and I think the GIMIX systems go continue the directory on the second side on track 0. My conjecture may be wrong, and I certainly wonder why the two main suppliers of FLEX based SS-50 bus systems couldn't get together! (I'll expect a call from Richard Don setting me straight, about the time this issue is out).

I do know for certain, that GIMIX FLEX formatted disks are compatible with FHL FLEX disks on the Color Computer all the way through double sided double density and 80 tracks. That fact was of great value in trying to get the software we purchased for a new CC transferred to 80 track disks compatible with the drives we bought for that system. That experience also answers a question that came up a couple of columns ago from a reader regarding running 80 track drives on the CC. The only problem I've found is that none of the CC FLEX implementations allow setting a drive to "double stepping" mode to make it possible to read the supplied 40 track disks. (I mean, of course, the way most software is supplied for CC FLEX). My procedure was to connect the two 80 track drives to a system running GIMIX FLEX and set one to the doub e stepping mode. Then I could copy 40 track disks to 80 track disks.

I have no SWTPC hardware that allows me the luxury of double density recording, so I can't verify the following conjecture, but I am fairly certain that SWTPC formatted disks will be compatible with DATA COMP FLEX on the CC. If Don Williams knows for sure, he will stick a note in here and indicate yes or no. •

PL9 AGAIN

I know, some of you are getting tired of hearing about this new compiler from Windrush. I'll give you a brief story and let it go. First of all, I need to report that Windrush has fixed all the bugs I know for sure to be real bugs in PL9. I have recently uncovered a few peculiarities, but they may well report to me that "that is how it works", and I won't be upset because the few remaining difficulties may be gotten around by writing the program a little differently.

Those of you who have my book "From BASIC to Pascal" will recall the text formatter program I did as an example of a larger program. Since that writing a couple of years ago, I've been using my program that I call JUST (for Justified text), for all my correspondence and work memos. The original in one version of Pascal produced just over 8K of object code. Another Pascal that I have would have produced 9 or 10K. The other day I decided to translate it into PL9. The program listing is about 10 pages of text, and the chore consisted of a lot of global edits and then a line by line edit to catch as many of the differences as possible. That took only a few hours. The debug took about four hours and several bug fixes before the formatter showed any major sign of life. Once I had output to the t minal going, it took anothe three hours to eliminate the remaining "gross" bugs. At that point, I had about 4.5K of object code.

In using the new JUST over the next couple days, I discovered a few more bugs and some ideas came to mind to reduce the code a bit. As of this evening, JUST is just over 3.5K. It runs several times as fast as the original, and the listing is reduced to 9 pages. In the rewrite, I made some of the constants (lines per page for example) and the default values for others such as line length, left margin, and paragraph indent what an assembler programmer would call "FCB's" at fixed memory locations. A user who receives the object code only, will be given a map of appropriate addresses where he can change the page length to 88, if for example his printer prints 8 lines per inch rathe than the more standard 6. The tailoring can be accommodated with a short program assembled and appended to the main file, or by loading JUST and changing the values in memory and saving it again.

JUST now outputs justified lines of text to the CRT just a tad slower than the LIST utility. The processing time has decreased significantly. Anyone who is familiar with Assembler programming but wants a higher level language, will find PL9 a very useful tool. "BASIC only" programmers may find that they have to dig a bit to understand and use PL9. I think it would be worth the effort for the writing of a "text file filter" program or perhaps a fast data collection program, to mention a couple of possibilities.

Along the line of fast and small, those of you who have been following this column for a while, will remember some compiler comparisons I did some time ago. The winner at that time (by a few seconds) was TSC's Pascal which executed my program to find P lme Numbers to a limit of 10000, (by means of an efficient divide algorithm and NOT a Sieve method) in 59 seconds. The PL9 version executed in 56 seconds. If you want something a shade faster, Windrush "C" ran it in 54 seconds.

Should you be interested in size of output code, the winne among all software I've tested for this program is OmegaSoft Pascal with about 2600 bytes. It executed the program in 67 seconds, which is in there with the fastest. Dugger's "C" and Windrush "C" were both just over 6K, but Windrush "C" executed the program in 54 seconds compared to 1:16 for Dugger's. PL9 generated 4.5K of output code for this program. I note here that OmegaSoft uses a library and a linking loader that only loads the modules used by the current program. It seems to generate minimum code for very small programs because it loads only minimum runtime routines, all of which are done in assembler. Some of the others, including Lucidata Pascal and PL9 generate considerably more code for small programs because their runtime code is not so modular, and in fact may be in the form of a library written in the language rather than assembler. This is the case for much of PL9 and the "C" compilers. For a very large program in the order of 12K or so, Lucidata pulls ahead by generating about 20% less code, and PL9 by about 30 to 40%. That is, the initial overhead is higher, but the code generated for user program is smaller.

I noted this peculiarity about compilers in previous discussions. Which one is most efficient is not just a straightforwa d question. The answer depends greatly on which features of the program you will be using, and on the size of your program. If you are writing programs to run in a 56K system, the question of program size is not really very important, but if you are designing stand alone control systems, the number of EPROMs per unit might be very important to the overall cost of the controller.

Still More

I continue to be a bit overwhelmed at the abundance of new software to run under FLEX. The new compilers continue to become available at an ever increasing rate. I have just finished a review of GSPL from Workman & Associates. I might as well throw in the statistics for it here too. It ran the same prime program in 55 seconds, and generated very nearly the same amount of code as PL9.

I notice a trend here. All these compilers are getting more efficient and faster, and the disparity between the execution times for the same program is shrinking. It would seem that the programmers are squeezing that last drop of performance out of the 6809 with their efficient coding, and we are approaching the

performance limit. Now you can sit back and take your choice of the language you like the best, and get an efficient compiler.

6809 - Z80

I just today received my June '68' MJ. I was pleased with the article by David Shearer comparing FLEX and CP/M, the 6809 and the Z80, and the respectively available software for the two systems.
I ought to diverge for a moment and tell you how I got into the 6809 initially. I had bought a KIM-1 board and it soon became obvious that the Microprocessor is the "wave of the future" for use in the sorts of things that I do for a living. I showed the company some sample programs that controlled lights and "relays" with pushbuttons (via I/O ports and computer programmed logic, of course). It soon became apparent that we would have to invest in a development system.

I called the Intel office in Cleveland, the closest one at the time, and the Motorola office in the Detroit area. I got a quick brush off from the Intel office with a "You'll have to spend $15,000 or so to get anything usable." By that time I had my first 6800 system from SWTPc running with a cassette interface and an old TTY, and I had invested about $600, so that seemed a bit steep. The Motorola office sent a couple of "engineer types" and they were most cooperative in getting us going on a shoestring. We eventually did spend about $15,000, but it was after we had seen some initial results, and we knew that we had some local support for our Motorola Exorciser system. So, you see, my getting involved with the 680X was more or less by chance.

Back to David's article. I have no disagreement with any of it. I was a bit surprised to see the number of good (read that "fast") compilers that were run in the CP/M tests. Since 2 MHZ is as fast as we can presently run our 6809, I guess it is fair to compare the Z-80 at 4 and 6 MHZ.

I certainly agree that the language implementation is more important than the language itself when looking for fast execution and efficient coding. Just looking at David's execution time summary, I would say that Lucidata Pascal's time must have been on a 1 MHZ system. I have generally found it to execute about twice as fast as TSC XBASIC, which would put it at 420 seconds. I got interested and coded the benchmark algorithm in PL9 and, since I had it in another C, made the easy transition (change one #include statement) and compiled it in Windrush "C" as well. Since neither of these were included in David's tests, you might be interested in the results on a 2 MHZ system. PL9 came in at 14 seconds, and Windrush "C" at 10.

I have personally coded a similar algorithm in Assembler, but for a different range of numbers. I've seen a sieve program in assembler that found the primes to a limit of 10,000 (1230 primes) in 0.14 seconds on a 1 MHZ system (for one execution, not 10). My algorithm timing tests were just for crossing out the non-primes in the array. I later did a count of the remaining non-zeros in the array for the prime count. If we have to compare on an equal basis with a processor running at 4 MHZ or 6 MHZ, then let's take the best crack at it and see if we can beat the reported 5.10 seconds for ten iterations while adhering to the test algorithm.

I started with my old program and the "C" version, and I believe the listing that accompanies this column accurately represents the algorithm, so that there were no "short cuts" taken. I used the registers as follows: U contains the variable I, D contains J and X contains K in the main loop (the one where all the time is spent). In this algorithm all three of those variables exceed a single byte value before the program is done, so no short cut could be taken such as using B and the ABX instruction. At any rate, I thought I would make the iteration loop count 100 so I could get accurate timing of the execution.

On my 1 MHZ system, 100 executions took 63 seconds, or 6.3 seconds for ten times through. The execution time on the 2 MHZ system at work turned out to be 3.3 seconds, or about 2/3 of the time reported in the article. I think the only "trick" used in the assembler program was to preload two array locations at a time using a pre decrement instruction on X. I spent an hour or so rewriting the program and getting it running. I am sure that someone will find a faster way to code the main loop. My first two tries at reducing the code resulted in about a ten percent improvement in execution time. The end result is 97 bytes of code, 5 bytes of "global variables" and 8K of data array.

As an aside, the listing PRIMC contains one small change in the algorithm, and it runs in 4.8 seconds for 10 iterations at 1 MHZ, or about 2.5 seconds at 2 MHZ.

The change comes from the realization that by the time we have crossed out all multiples of primes up to the square root of 16384, there are no more to cross out. That happens when I reaches the value of 64. After that point, we only have to scan the array and count the remaining non-zero locations. In order to implement that better, I moved the count of primes up to the loop that looks for the next non-zero array location. I know that PRIMC is an improvement on the algorithm and is therefore "cheating". Someone out there improve the coding for PRIMEB a bit more and we will have really impressively beat the Z-80 running at 6 MHZ with our 2 MHZ processor. Motorola, how about a 3 or 4 MHZ 6809???

There is another point to this. If we can beat the Z-80 in Assembler code, there is no reason why compilers in 6809 can't beat the Z-80 compilers. As I indicated above, there is a trend in the direction of more efficient and faster output code from some of the most recent compilers for the 6809. If it is indeed true that we have a smarter and better instruction set, as our impartial reviewer indicates in his article, it should be possible to write compilers that generate faster and more efficient code with those instructions. I don't think the Z-80 compiler writers are any smarter. Perhaps it is just that their market is bigger so they can spend more money on the development of their products? David, it would have been interesting if you could have indicated the total bytes of code generated by each compiler for the benchmark program.

David mentioned the multiplicity of sets of mnemonics for the Z-80. We 6809 users don't have anything quite that bad to cope with, but we do have differing sets of pseudo operation codes in the many assemblers that are available. Things seem to be pretty standard until we get into relocatable assemblers. Then Motorola uses XDEF to flag a label that is defined externally from a module, and XREF for one within the module that is to be referenced from another module. Hemenway used ENT and EXT for his 6800 assembler (EXTernal and ENTry, I suppose). Though most of our assemblers use $1000 to indicate HEX value 1000 (4096 decimal), there are a few maverick assemblers that use 'X1000 to mean the same thing. There are also variations on the FCB, FDB, and FCC directives.

I was interested in David's conclusion that FLEX is better documented than CP/M for someone who is building his own system and has to write his own disk drivers. I had heard an exactly opposite opinion from a reader who wrote me a few months back.

David, I agree that the CP/M group has much more choice of software. There's bad, mediocre, average, and excellent software available both for CP/M and for FLEX. If there is more available for CP/M, I have to assume that there is more in each category.

Regarding my previous remarks about Wordstar, I realize that it was around first, and that everyone who wrote a text processor after that time, has copied the good parts and improved the weak ones. Such is progress. I'm afraid I might have over reacted to someone who told me how GREAT Wordstar was, and how it could do everything. This person had never seen anything else and had no basis of comparison when he recommended it to another friend as the BEST. Ulterwyk's BASIC was the BEST we early users of the 6800 had. It was and still is an excellent implementation of BASIC, but it is VERY slow compared to what we have available today. (I would guess it might come close to beating Cobol on a 4 MHZ Z-80). Stating the fact that neither Wordstar nor Ulterwyck BASIC is presently the best, doesn't mean that they have deteriorated, just that progress has been made in the meantime.

David, thanks for a fine and thoughtful article. We need more such informative articles by people who have the firsthand information.

TRIVIA SIDE

Maybe some of my readers consider this sort of stuff a total waste of space, but I've been chuckling for a couple of weeks. You know how I complain about manuals not quite saying what they mean.. The other day I was looking at a box of Arm and Hammer Baking Soda. Right there on the side panel it said "BRIGHTENED SMILES -- Pour some soda into your hand and brush with wet toothbrush." I don't quite see how having a handful of

wet baking soda will brighten my smile. Did I miss something? I'd say that I didn't, but the poor fellow who wrote that neglected to tell me that he meant for me to use the wet brush on my teeth! Come to think of it, it worked without even getting that soda in my hand. I've been smiling over it for some time.

• Editor's Note: Yes, Data-Comp F-MATE FLEX™, as well as FHL FLEX™, is happy with any GIMIX™ format. F-MATE FLEX will also click right along with SWTPC single density, single sided, 40 track disk formats.
Data-Comp will release, in the near future, double stepping for their FLEX adaption. It should be retrofitable to their earlier versions.

DMW - - -

```
 NAM PRIMC
 TTL PRIMES BY SIEVE
 OPT PAG
 PAG
*
* FILE PRIMC.TXT IMPROVED ALGORITHM
*
* PRIMES BY SIEVE IN ASSEMBLER FOR 6809
*
* BENCHMARK ALGORITHM FROM BYTE
*
* FLEX EQUATES
*
OUTDEC EQU $CD39 OUTPUT DECIMAL NUMBER
PSTRNG EQU $CD1E FLEX PRINT STRING ROUTINE
PCRLF EQU $CD24
WARMS EQU $CD03
*
SIZE EQU 8190
*
* THESE ARE DIRECT PAGE VARIABLES
 SETDP $30 ASSEMBLER DIRECTIVE
 ORG $3000
COUNT RMB 2  PRIMES COUNT
LCOUNT RMB 1 LOOP COUNT
*
*
 ORG 0
 RMB 8192 FOR PRIMES TO 16381
PREND EQU *
*
*
START LDA #$30
 TFR A,DP SET DIRECT PAGE REGISTER
 LDA #100
 STA LCOUNT
LOOP LDI #SIZE+2
 LDD #$FFFF
*
* LOOP TO INITIALIZE ARRAY
*
FILL STD ,--I
 CMPX #1 WILL GET TO 1 OR 0
 BHI FILL
*
* FIND NEXT NON-ZERO PRIME
*
 LDY #0 COUNT OF PRIMES
 LDU #0 THIS IS THE INDEX VARIABLE "I"
NONZ TST ,U+ REPEAT UNTIL PRIME (I) <> 0
 BEQ NONZ
 CMPU #SIZE+1
 BGT DONE
 LEAY 1,Y COUNT PRIME HERE **********
 CMPU #66  **********
 BGT NONZ  DON'T NEED TO CROSS OUT ANY MORE, JUST COUNT
 LEAU -1,U FOR THE POST INCREMENT
*
* MAIN LOOP
*
MAIN TFR U,D
 TFR U,X
 ASLB
 ROLA
 ADDD #3 HAVE INITIAL J
 LEAX D,X X CONTAINS K
CRSLOP CMPX #SIZE  THIS IS THE "CROSSOUT" LOOP
 BGT NXTNDX WHILE K <= SIZE
 CLR 0,I
 LEAX D,X K=K+J
 BRA CRSLOP
NXTNDX LEAU 1,U  I=I+1
 BRA NONZ
DONE STY COUNT
 LDX #COUNT
 JSR OUTDEC
 LDX #MSG
 JSR PSTRNG
 DEC LCOUNT
 BNE LOOP
 JMP WARMS
MSG FCC / PRIMES/
 FCB $0D,$0A,$04
 END START

 NAM PRIMB
 TTL PRIMES BY SIEVE
 OPT PAG
 PAG
*
* FILE PRIMB.TXT IMPLEMENTATION OF BENCHMARK ALGORITHM
*
* PRIMES BY SIEVE IN ASSEMBLER FOR 6809
*
* BENCHMARK ALGORITHM FROM BYTE
*
* FLEX EQUATES
*
OUTDEC EQU $CD39 OUTPUTS 16 BIT NUMBER POINTED AT BY X
PSTRNG EQU $CD1E FLEX PRINT STRING ROUTINE
PCRLF EQU $CD24
WARMS EQU $CD03
*
SIZE EQU 8190
*
* THESE ARE DIRECT PAGE VARIABLES
 SETDP $30 ASSEMBLER DIRECTIVE
 ORG $3000
COUNT RMB 2  PRIMES COUNT
LCOUNT RMB 1 LOOP COUNT
INDEX RMB 2  "I"
```

```
*
*
 ORG 0
 RMB 8192 FOR PRIMES TO 16381
PREND EQU *
*
*
START LDA #$30
 TFR A,DP SET DIRECT PAGE REGISTER
 LDA #100
 STA LCOUNT
LOOP LDX #SIZE+2
 LDD #$FFFF
*
* LOOP TO INITIALIZE ARRAY
*
FILL STD ,--X
 CMPX #1 WILL GET TO 1 OR 0
 BHI FILL
*
* FIND NEXT NON-ZERO PRIME
*
 LDY #0 COUNT OF PRIMES
 LDU #0 THIS IS THE INDEX VARIABLE "I"
NONZ TST ,U+ REPEAT UNTIL PRIME (I) <> 0
 BEQ NONZ
 CMPU #SIZE+1
 BGT DONE
 LEAU -1,U FOR THE POST INCREMENT
*
* MAIN LOOP
*
MAIN TFR U,D
 TFR U,X
 ASLB
 ROLA
 ADDD #3 HAVE INITIAL J
 LEAX D,X X CONTAINS K
CRSLOP CMPX #SIZE  THIS IS THE "CROSSOUT" LOOP
 BGT NXTNDX WHILE K <= SIZE
 CLR 0,X
 LEAX D,X K=K+J
 BRA CRSLOP
NXTNDX LEAY 1,Y INCREMENT PRIME COUNT
 LEAU 1,U  I=I+1
 BRA NONZ
DONE STY COUNT
 LDX #COUNT
 JSR OUTDEC
 LDX #MSG
 JSR PSTRNG
 DEC LCOUNT
 BNE LOOP
 JMP WARMS
MSG FCC / PRIMES/
 FCB $0D,$0A,$04
 END START
```

COLOR User Notes

Robert L. Ney
5900 Cassandra Smith Rd.
Hixson, Tn. 37343

A Color Computer Disassembler

Many "moons" ago, I reviewed the excellent Micro Works Disassembler, which was one of the first available for the Color Computer. A few others have appeared over the past couple of years, but they are usually written in BASIC, and do not provide the features that make the Micro Works Program so useful. But, it also has a few weaknesses. The primary problem is that there is no easy way to obtain the Source Output in a "Computer Usable" form (so it can be edited, etc.), other than with their SDS-80 Assembler. Even then, there are no Disk capabilities available. In fairness, it was written before the Disk Systems appeared, and it is primarily oriented towards disassembling the BASIC ROMs; and that it accomplishes more than adequately. It was still one of the few Disassemblers that provided a "true" disassembly of Offset Code before CoCo SLEUTH appeared.

Well, at last, there is a REAL Disassembler for the Color Computer! The FLEX Users have had access to Computer Systems Consultants, Inc. "Super Sleuth" for a few years; it is now available for the Color Computer Users.

CoCo SLEUTH
Computer Systems Consultants, Inc.
1454 Latta Lane
Conyers, Ga. 30207
Requires 32K RAM and Disk System -- $49.00

CoCo SLEUTH is a Machine Language Program which requires a 32K Color Computer (or "look alike") and a Disk System (one Disk Drive will work fine). The CoCo SLEUTH Disk actually contains three (3) different programs; SLEUTH/BIN (the Disassembler), CHGNAM/BIN (a program that allows you to replace SLEUTH generated labels, such as "ZC004", to a named label such as "DSKCON"), and XREF/BIN (a cross reference generator). CoCo SLEUTH circumvents the Display limitations with a 51x24 Display Screen, and is an EXTREMELY Powerful, Easy to Use, INTERACTIVE Disassembler. If that is not enough, it will Disassemble 6800, 6801, 6802, 6803, 6805, and 6502 Binary Code (if you can get the Files on a Disk); Disassemble a Disk Binary File and write the Source Code to another Disk File in "Assembler Source Code Format", AND provide a "Full" listing to the Screen and/or Printer; Disassemble a program in Memory and do the same things with the output; Dump a Binary File or Memory Resident (RAM OR ROM) object code to the Screen in a memory-dump format, allow "Full Screen Editing" modifications, and write the modified code to a Disk File; etc.; all without disturbing the ORIGINAL Code on the Disk or in Memory. There is still MORE! A "position independent switch" is provided which allows converting the 68xx Series Chip Codes to position independent code for the 6809 (sorry, it is not "fool proof", you will have to check it carefully, and possibly make some more manual changes). Very large Programs (BASIC??) can be broken into smaller parts so the Source will fit on the Disks. The Source Code from the non-6809 Chips can be output so that the Computer Systems Consultants Cross Assemblers can be used with a Macro Assembler to be reassembled in that Chips' Binary Code (i.e., you can use CoCo SLEUTH to disassemble 6502 Object Code, make changes, and reassemble it back to 6502 Code, ALL ON THE COLOR COMPUTER). As you can see, this is a potent package!

Before we get into the operation of CoCo SLEUTH, let's look at Disassembly methods in general. Normally, you run a "preliminary" disassembly of a Binary File, or area of Memory, with an "ASCII Flag" turned on so you can pick those areas out of the Disassembly. Why pick them out? The Disassembler does not follow the CPU's normal flow through a Program, so it does not know what portions of the Program are to be used for what purpose; it simply tries to interpret everything as "Instructions". Since Data Areas are not intended to be Instructions, the Disassembler gets all tangled up. The "good" ones do a pretty good job of getting reoriented and "back on track", but there is still going to be a lot of confusion in the Output Listing. You study this initial listing to locate things like the program Logo, Error Messages, Reserved Words, etc., anything that is obviously Characters and not Instructions. You then tell the Disassembler where these areas are, and make another pass (the ASCII, or "FCC", areas are fairly easy to locate and identify). Now, you will have a "cleaner" output because the Disassembler knows that those areas are

not Instruction Code, and outputs them as they would be written in Source Code. The real problems appear in trying to identify things like Tables, Jump Tables, Data Areas, etc. You have to study the Code to see what is what, identify more areas for the Disassembler, run another Disassembly, etc. (This is also an EXCELLENT way to learn "How to Program"; you will learn a lot from studying the Disassembly Listings.) **CoCo SLEUTH**, and the FLEX Based **Super SLEUTH**, radically simplify this procedure by providing an **INTERACTIVE** environment for this "Definition" phase of the Disassembly.

CoCo SLEUTH is started with the normal Radio Shack Machine Language LOADM"SLEUTH:EXEC routine; you can then remove the Disk because the Program is Memory Resident, and insert the Disk containing the Binary File to be worked on. You get a 51 Column by 24 Row Display of Black Letters on a Green Background, and a Program Intro Logo followed by the **CoCo SLEUTH** question mark prompt. You can enter a "?" any time you have the question mark prompt and receive a "Help Menu" in the form of a listing of the Commands with a note on their use.

When SLEUTH is up and running, hit as "S" and it will request a "filename"; answer this with something like "PROG/BIN", and SLEUTH loads the File and gives it's starting and ending addresses, transfer address (or EXECute address), and informs you of the "state" of the Disassembler, such as "separate-label switch off", "cpu mode 6809", "position-independence switch off", and "cross-assembler switch on". Any of these can be changed at will, but this is the normal operating environment. Now that the File to be Disassembled is loaded, hit a "Q" and you get a "Dump" of the Object Code on the Screen.

The "Dump" Format displays 128 Bytes of Code in 8 Rows of 8 2-Byte Columns, 8 ASCII characters for the Bytes, and an "Info" Column on the right side as shown (the "xx" represent valid Hex Bytes - this IS a Copyrighted Program from **MTP, INC.**).

```
     .1 .2 .3 .4 .5 .6 .7 .8  01234567
0F00 20#18#01#00#08#28#43#29#  ....(C) start
0F08 20#31#39#38#32#20#42#59#  1982 BY =0F00
0F10 20#4D#54#50#2C#20#49#4E#  MTP, IN end
0F18 43#2E#12 17 04 45 CC C0  C....EL@ =161A
0F20 00 10 83 44 48 26 03 BD  ...DH&.= a=!
0F28 xx xx xx xx xx xx xx xx  J;=)(0.. c=#
0F30 xx xx xx xx xx xx xx xx  q..JL .. h=(
0F38 xx xx xx xx xx xx xx xx  .L'.Z&.. i=)
0F40 xx xx xx xx xx xx xx xx  .7..0... j=*
0F48 xx xx xx xx xx xx xx xx  ..3F...L k=+
0F50 xx xx xx xx xx xx xx xx  -. .'z.. scrn
0F58 xx xx xx xx xx xx xx xx  '...'.A. quit
0F60 xx xx xx xx xx xx xx xx  $n-. .'. prev
0F68 xx xx xx xx xx xx xx xx  @ e)'b-. next
0F70 xx xx xx xx xx xx xx xx   .. Z'.  addr
0F78 xx xx xx xx xx xx xx xx  Ww.Q.BI?  ?
```

As you can see, there is a lot of information on the Display. The far right Column of Information shows the start and end addresses of the File or Memory Area to be Disassembled, the "Defined Area Keys" for the display, and the Commands available to the User in this mode of operation. It is easy to see that there is an area of ASCII characters near the beginning of this Program, and that the first three Bytes have been defined as "Instruction Code" with the ASCII Area defined as "Code" or an area of FCC. The Cursor is setting at the bottom right of the Display, following the bottom "?". A "Q" for "quit" was entered when the Display originally came up, and the "?" prompt was displayed on the line below the Display (the full Display is still visible). That prompt was answered with a "C" to let CoCo SLEUTH know that we wanted to define a Character, or FCC, area, and answered the address prompts with 0F04 and 0F19, defining the beginning and ending of that area. Now, if a "Q" was entered again, the "Dump" is redisplayed, except there will be a "#" in the space following the Bytes within that area as shown above. You can "page" through the area to be Disassembled by entering an "N" for the "next" page, or back by entering a "P" for the "prev"ious page. If you want to Jump to another page, you can enter a Hex Byte for the "addr"ess command. If you entered "12", the Display would begin with address $1200.

If an "S" is entered, the Cursor goes to the Top Left corner of the Display for the "Screen Edit" mode. The Cursor can be moved anywhere with the Arrow Keys, and the Hex or ASCII is changed by simply typing new information at the Cursor location. You can use this mode of operation to modify ROM Code by dumping the ROM, making changes, and then "quit" the Dump Mode and "Write" the modified code to a Disk File. As was indicated earlier, CoCo SLEUTH does NOT change the ORIGINAL Code on a Disk File or in Memory (ROM Code is pretty hard to modify, anyway). It stores the changes in internal "Tables", and makes the changes as the Code is written out to Disk.

Back to the Disassembly of the Program. Suppose we have now defined the obvious FCC Areas, and want to see if there are other fairly obvious areas to be found. After "quitting" the Dump Mode, we can enter the "Y" Command for an **ABSOLUTE Disassembly** of the Code. This is a simplified Disassembly that does not use Labels; all branch references are shown as the ADDRESS itself. This, again, makes it easy to see things like LDX **$12FC,PCR** which may indicate that Reg. X is being used as a pointer to a Data Area. If we follow the code a little more, we can make this determination, and we have found another "Area".

Every time we define or change an Area Specification, it is "logged" internally within **CoCo SLEUTH**. If this is a fairly large Program, it will take a WHILE, and we may have a dozen different areas defined. Any time you want to see what the "Areas" and "States" of CoCo **SLEUTH** are, you can enter an "L" and it will be listed on the Screen. Also, you can SAVE all of this "hard work" to a Disk File, so you don't have to start all over again later.

Since most of you are fairly familiar with normal Disassemblers and their output, I will only say that CoCo SLEUTH provides all of the normal Output Options, PLUS! As mentioned previously, you can obtain an output that is ready for a normal Macro Assembler if you are working with other than 6809 Code, along with obtaining Disk Files ready for a 6809 Assembler so that you can make changes and Assemble the code for a different program, etc.

I stressed the **INTERACTIVE** nature of CoCo SLEUTH (ALL of these features are common to the original FLEX Based SUPER SLEUTH also) at the beginning. You can now see what I meant. **CoCo SLEUTH** is an absolute **MUST** for anyone that is working with Assembly Language, either in LEARNING it, or USING it. It is obvious that it was developed by someone that has been working with Assembly Language for a long time, in that the features found in this Program are just not available anywhere else. At $49.00, **CoCo SLEUTH** is the "Steal of the Century" for Color Computer Users.

--- RLN ---

"C" User Notes

Norm Cosmo
3 Pryor Road
Natick, MA 01760

This month will start a three column series that will introduce you to writing device drivers C. We'll cover the 6850 ACIA and the 6821 PIA since they're the two types of I/O device we all have in common. This column will cover the basic drivers for a polled I/O environment. The second column will look at a very simple IRQ scheme. The third column will cover buffered data handling with interrupt driven I/O.

I will also be reviewing a spate of recently released C compilers. This month it's the Windrush compiler, next month will be Microware's compiler, then Dynasoft's and finally I would like to cover the Everhardt, or PDS-C, compiler available from Word's Worth. Somewhere in all of this it would be nice to actually get in some programming in the C language to boot!

In the future, I hope to find a way of getting the Everhardt compiler and the Ream "Small C" screen editor published here. These are two programs that teach a lot about programming and the C language. The Ream editor was published in Dr. Dobb's a few months ago. It is a nice screen oriented line editor. I have a version that one of the compiler vendors sent me. It ran with no playing at all other than the terminal handlers for my H19 terminal.

I/O DRIVERS

C has often been described as a middle level language, falling somewhere between a "real" high level language and assembler. As such one of its strong points is the ability to interface to, and control, the hardware it's being run on. Indeed, I've read in more than one article how the basic UNIX operating system consisted of about 20,000 lines of C code supported by only 800 lines of assembler. That's a pretty impressive ratio.

In our case, other than the size limitation imposed by 64K, we can write very impressive applications for stand alone hardware systems where assembler is used only to get the system started and for handling

interrupts. There are several mechanisms for addressing hardware in C, all based on pointer types. These are the pointer, arrays and structures. So let's dive into it.

The first problem is how we'll model multi-register ports. Since most of our ports are 8 bit devices, some sort of pointer to char's is the natural choice. It turns out that arrays and struct's can really simplify the task.

AN ACIA DRIVER

Consider the 6850 acia. It has four registers that take up two sequential addresses. We could define a pointer to a character and then set it equal to the base address of the device as shown in the following character input function.

```
serin()
    {
    char *port;

    port = 0xE004;
    while ((port & 0x01) == 0)
        ;
    return(*(port+1));
    }
```

This is a rather cumbersome way to handle the problem, for a number of reasons, but it would certainly do the job of getting characte s from the console. The single semicolon after the while statement is really a NOP and causes the program to stall in the loop until there is data ready.

The most obvious thing to do is to make it more general by passing it the port address. Then we can use it for getting data from other ports, such as a modem. Next, we can define some symbolic constants to add in the necessary offsets for the data and status/control registers. Finally we would also define symbols for the various status bits. We might have a define section that looked like.

```
#define TERM 0xE004
#define MODEM 0xE000
#define STATUS 0
#define CONTROL 0
#define DATA 1
#define RDRF 1
#define TDRE 2
#define NOTREADY 0
```

Combining all these ideas, we could rewrite the function as

```
serin(port)
    char *port;
    {
    while(((port+STATUS) & RDRF) == NOTREADY)
        ;
    return(port+DATA);
    }
```

The function is now more useful and more readable. The compiler will add the offsets to the pointers with no problem. But now let's define the port as an array of characte s and see what it looks like.

```
serin(port)
    char port[];
    {
    while((port[STATUS] & RDRF) == NOTREADY)
        ;
    return(port[DATA]);
    }
```

This is a much cleaner way of handling multiple register devices like an acia, pia or timer chip. ,pp
We could have made the source code tighter by writing

```
while((port & RDRF))
```

since we know the status port has a zero offset from the base address. That will save a (very) few bytes of code. Using the logical inverter in this case saves us a few bytes of file size but probably renders no savings whatever in the binary. But let's be honest, it is not as readable as the previous fo m.

In the program, the function would be called with a line similar to

```
c = serin(TERM);
```

If your compiler supports structs, then you could define a struct that describes each type of device. For an acia we might use this.

```
struct acia {
        char status;
        char data;
        };
```

Rewriting serin() would yield

```
serin(port)
    struct acia *port;
    {
    while ((port->status & RDRF) == NOTREADY)
        ;
    return(port->data);
    }
```

The call to serin() would be the same as it was in the previous example. Personally, I prefer using struct's but I doubt that there is any real savings in code. To me it just looks cleaner. If you don't have struct's then treating device as an array of chars is the cleanest.

It is interesting that when using the array form a particular register can have a number of different names; they are all merely defined as the same offset from the base address. However, if you are using struct's it gets dirtier since you must use a union buried within the struct. The definition might look like

```
struct acia {
        union foo   {
                    char status;
                    char control;
                    };
        char data;
        }
```

The while loop would look like

```
while ((port->foo.status & RDRF) == NOTREADY)
```

That is clearly not as readable as the first example. Maybe it's better just to call it status and consider it read/write.

For the rest of the discussion, I will be using the array of char's for since it can be used by any of the compilers that are available today.

Now let's write a complete package. We will need to initialize the device and handle both input and output. it would also be nice to be able to actually check whether or not the port is ready for reading or writing. So we will need six functions called init 6850(), serin(), serist(), serout(), soutd() and serost(). These functions will use the following #define's.

```
/*
 * hardware addresses and register
 * offsets.
 */
#define TERM 0xE004
#define MODEM 0xE000
#define SCONTROL 0
#define SSTATUS 0
#define SDATA 1
#define RDRF 1
#define TDRE 2
#define IRQBIT 0x80

/*
 * control and setup values
 */
#define RESET 3
#define B300 0x11
#define B1200 0x12
#define LF 0x0A
```

You might want to take these and put them in a header filed call, say, ACIA.H. Note that the constants for the baud rate are based on the acia being wired for 1200 baud with the internal divider set to 16. In this way, 300 baud can be obtained merely by setting the divider to 64. You might want some other values if your system uses parity, interrupts, etc.

The initialization routine will be passed both the port's address and its initialization value so that it can be used for any acia.

```
init_6850(port,setup)
    char port[], setup;
    {
    char junk;

    port[CONTROL] = RESET;
    port[CONTROL] = setup;
    junk = port[DATA];      /* clear the flag */
    }
```

For input, we want a function that returns TRUE if data is available, FALSE if it isn't.

```
/*
 * check the status of the input,
 * return TRUE if data is ready,
 * FALSE otherwise.
 */
serist(port)
    char port[];
    {
    if ((port[STATUS] & RDRF) == NOTREADY)
        return(FALSE);
    else
        return(TRUE);
    }
```

Note that if this function were to be used only with ASCII data and not binary, then it could return the character to indicate TRUE. The only collision would be with NULL, but who cares. This way might be handy in certain circumstances.

The function serin() now changes a little bit from the previous example.

```
/*
 * get a character
 */
serin(port)
    char port[];
    {
    while (serist(port) == NOTREADY)
        ;
    return(port[DATA] & PARMSK);
    }
```

If the function is to be used for binary data, then remove the parity stripping or you'll get into trouble.

The functions for output data would be quite similar, with similar caveats applied if you want to use them for handling binary data.

```
/*
 * return TRUE is the xmitter is
 * ready for data, FALSE otherwise
 */
serost(port)
    char port[];
    {
    if ((port[STATUS] & TDRE) == NOTREADY)
        return(FALSE);
    else
        return(TRUE);
    }


/*
 * output a character, expanding any
 * newlines to <lf><cr>
 */
serout(c,port)
    char c, port[];
    {
    if (c == '\r')
        soutd(LF,port);
    soutd(c,port);
    return(c);
    }


/*
 * output a character
 */
soutd(c,port)
    char c, port[];
    {
    while (serost(port) == NOTREADY)
        ;
    return(port[DATA] = c);
    }
```

A PIA DRIVER

The 6821 consists of four basic registers; but for our purpose, we will look at it as a dual two register device. The ways in which a pia can be used are many. To limit our discussion, we will treat it as a parallel port printer driver of the "Centronics" variety.

We're smarter now, since we've already designed the acia drivers, so we'll immediately define some useful constants.

```
#define PRINTER 0xE00A      /* my system only */
#define PDATA 0
#define PCONTROL 1
#define PSTATUS 1
#define PWR BITS 0xFF
#define PSETUP 0x2C
#define PRDY_BIT 0x80
```

Since the printer is output only, we will only need four functions, called Init_6821(), parout(), poutd(), and perost().

THe initialization for the 6821 is straight forward; once you figure out how to set up the state of the pia. This one really tripped me up for a while. Quite frankly, the Everhardt complier arrived just in time so I lifted the routines from his library.

```
/*
 * init the pia
 */
init_6821(port,wbits,setval)
    char port[], wbits, setval;
    {
    port[PCONTROL] = 0;
    port[PDIRECTION] = wbits;
    port[CONTROL] = setval;
    port[PDATA] = '\0';
    }
```

The routines to read the status and transmit the data are quite parallel to the serial drivers. One minor difference is that with the pia it is necessary to clear the ready flag by reading the port (even though we are sending data) since the busy signal from the printer goes to the CA(B)1 line, which is edge triggered.

```
/*
 * return TRUE if the port is ready
 * for data, FALSE otherwise
 */
parost(port)                                    )
    char port[];
    {
    if ((port[PSTATUS] & PRDY_BIT) == NOTREADY)
        return(FALSE);
    else
        return(TRUE);
    }



/*
 * print out a character expanding
 * a newline into <lf><cr>
 */
parout(c,port)
    char c, port[];
    {
    char junk;

    while (parost(port) == NOTREADY)
        ;
    junk = port[PDATA];
    port[PDATA] = c;
    return(c);
    }
```

As Bugs Bunny would say "ttttthat's all folks." We're now ready to compile the functions and put them to work. This month's listing is a simple exerciser for the functions. By making the appropriate address changes, it should fly on your system with no trouble. It may even have some utility if you want the VERY barest modem program. The program will turn your computer into a simple terminal ready to use on the modem.

THE WINDRUSH C COMPILER

I was really excited when this package arrived without any advanced warning. I had been reading the Windrush ad in 68Micro f r sometime with interest.

The package, which sold for $295 as of the May issue, includes the compiler (consisting of a number of passes), a special linker that works with the TSC relocating assembler, the standard library, some runtime code, a few header files and the primes test program (both source and compiled). Not included, but needed is the TSC Relocating Assembler, which c sts around $150.

The package arrived containing an the manual and a single 8" disk. Unfortunately, I run only 5" disks. So off to Georgia for recopying it went while I read and reread the manual.

I got back two 40 track disks, just about full. By proper partitioning I was able to make two bootable disks with the compiler passes, some utilities and the TSC editor. On to the details.

The Windrush compiler was written by James McCosh who also wrote the SWTP compiler that runs under Uniflex. The compiler is just about complete and lacks only floats, doubles, bit fields and a few of the least used compiler directives. The manual is quite concise but very clear and plainly points out the differences between the Windrush compiler and the standard language described by Kerhighan and Ritchie.

The compiler is designed to work with the TSC Relocating Assembler (SP09-17) mentioned previously (available from South East Media - see Advertising), although the assembler doesn't come as part of the package. I might add as a side comment that you owe it to yourself to buy the assembler/linker/librarian combo for its own sake. It is really a pleasure to be able to have REAL libraries and do all the linking at the binary level.

The compiler has a number of switches to control the compilations. The simplest case is

```
CC filespec.c
```

This command will do it all, producing an executable binary called filespec.cmd. Note that the file extension was used. Indeed it must be used so that the compiler can do some nice things for you. The default extension are ".c" for C source files, ".a" for assembler source files and ".r" for assembled, relocatable modules. Now consider the command line

```
CC file1.c file2.a file3.r
```

The compiler will first compile and assemble file1. Then it will assemble file2. Finally, it will link file1, file2, file3 and the standard library clib.lib t gether into and executable called OUTPUT.CMD. If there are no errors expected, then you can walk away for nice tall cool one.

A number of the usual s rt of switches are available such as

```
+a  don't assemble the module
+c  imbed the C code in the assembler code
+d= define a symbol
+f= explicitly name the output file
+l= include a library file
+n  no assemble code (quick syntax check)
+o  run the optimizer
+r  do not link the module
+s  disable stack checking code
```

The "=" means that an additional parameter must also be included. I found that the compiler ran smoothly which was comforting since I had to stop midstream to change disks. I use the +ro options since I must change disks for a complete compile, but once I forgot. The process stopped with a "NOT FOUND" message from FLEX when it tried to chain to the linker; but everything was fine. I changed disks and continued. Nice.

The code produced by the compiler is quite tight. The primes program ran in a little under 17 seconds on my 1Mhz system. That's reasonably fast. I have looked at the code and found it to be clean although you can certainly tell it was produced by a compiler. Unfortunately, the compiler does not keep the assembler source code around after it has been used. I tried using the +ao options, but the optimizer would never be invoked s I couldn't give you an example of the same code with and without optimizing.

As I said earlier, the manual was well done but would never win any prizes for verbosity. It starts by telling you what files you get on the disk, the features of the compiler, the compilations options, language extensions and compiler limitations. They do it all on two pages! Actually, it's not really that bad, they do amplify what needs to be. A really nice touch was four pages of diagnostic (error) messages and what most likely caused the them. That is really helpful.

There is a section on using the C loader stand alone. They list all the options and give some examples. But the one thing they didn't do was tell you explicitly

how the compiler runs it. The same for the assembler. It would be very nice to have a list of the default options used by the compiler as it calls in the various passes.

The standard library is well explained using the page format that has become a sort of default for Unix system manuals. It includes brief descriptions of the function, how it is used, how it handles errors and any necessary warnings or known bugs. The library is quite extensive. The library file, clib.lib, is called automatically and is expected to be on the system disk.

The "extension" to the language are the #asm and #endasm directives that allow you to put assembler code directly into a C program.

There were some minor and not so minor nuisances though. The first is a problem with the direct page register. They do not explicitly set it to where they want it. I surmise that they have assumed that FLEX sets to zero. Surprise. Not all (if any) do it. When I tried to run the compiler after running STYLO the system would invariably hang but the symptoms would vary. STYLO exits with DP set to $59.

The next problem area was in the treatment of the newline character, '\n'. In the standard it is defined as <lf>, which is the Unix newline. But FLEX uses <cr> which is '\r'. Windrush chose to make both '\n' and '\r' the same. I feel that this is wrong. By all means, do a conversion in the I/O routines if necessary, but don't make them synonymous. I would like to point out though that they claim programs generated under Unix will run under FLEX when recompiled with their compiler.

I would urge them to make one of the switches of the cloader available to the compiler. It would be very handy to have a symbol map for debugging without having to run the cloader stand alone.

What would will I give it? It's a very nice package, well thought out. I haven't use it widely, but I did do this column's programming with it. It runs smoothly and is fairly quick. The compiler implements almost all of the language and produces good tight code. Finally, there is a good support library. That much deserves an AAA. But there was the problem with the DP. If I hadn't seen the same bug already with Dugger, I would have kept me guessing for a while. The equivalence of '\n' and '\r' may be nit picking but it bothers me. I can't see dropping off a complete "A" so I'll stick with the AAA, but with some reservations.

IT'S A WRAP

That's it for this month. Next time we will cover handling interrupts in C, with a rewrite of this month's listing to have interrupt driven input. And there will be a review of the Microware compiler. Till then...

- - -

```
/*
 * drivers.c
 *
 * A test program for terminal/modem and
 * printer dirivers.
 *
 */

#include "I.STDIO.H"

/* serial stuff */
#define MODEM 0xE000
#define TERM 0xE004
#define SSTATUS 0
#define SCONTROL 0
#define SDATA 1
#define RDRF 0x01
#define TDRE 0x02
#define 8300 0x11
#define B1200 0x12
#define TERM_INIT 0X11
#define RESET 0x03

/* parallel stuff */
#define PRINTER 0xE00A
#define PDATA 0
#define PDIRECTION 0
#define PSTATUS 1
#define PCONTROL 1
#define PWR_BITS 0xFF
#define PSETUP 0x2E
#define PRDY_BIT 0x80

/* misc defines */
#define CNTL_C 0x03
#define CNTL_P 0x10
#define PARMSK 0x7F
#define LF 0x0A
#define READY 1
#define NOTREADY 0

main()
    {
    char c;
    int direction;

    init_6850(TERM,TERM_INIT);
    init_6850(MODEM,B1200);
    init_6821(PRINTER,PWR_BITS,PSETUP);

    direction = 0;
    FOREVER
        {
        if (serist(TERM))
            {
            c = serin(TERM);
            if (c == CNTL_C)
                exit();
            if (c == CNTL_P)
                {
                direction = ~direction;
                continue;
                }
            if (direction == 0)
                serout(c,MODEM);
            else
                parout(c,PRINTER);
            }

        if (serist(MODEM))
            {
            c = serin(MODEM);
            serout(c,TERM);
            }
        }
    }


/**************************************
*                                     *
* here are all the serial drivers     *
*                                     *
**************************************/


/*
 * init a 6850 acia
 */

init_6850(port,setcode)
    char port[], setcode;
    {
    port[SCONTROL] = RESET;
    port[SCONTROL] = setcode;
    }


/*
 * get a character, stripping parity
 */
serin(port)
    char port[];
    {
```

```
    while (serist(port) == NOTREADY);
    return(port[SDATA] & PARMSK);
    }


/*
 * test the receiver for data
 */
serist(port)
    char port[];
    {
    return(port[SSTATUS] & RDRF);
    }


/*
 * print a char to the port, expanding
 * any newline to (cr)(lf)
 */
serout(c, port)
    char c, port[];
    {
    if (c == '\r')
        soutd(LF, port);
    soutd(c, port);
    return(c);
    }
```

```
/*
 * low level acia driver
 */
soutd(c, port)
    char c;
    char port[];
    {
    while (serost(port) == NOTREADY);
    return(port[SDATA] = c);
    }


/*
 * test if the acia is ready for
 * xmit data
 */
serost(port)
    char port[];
    {
    return(port[SSTATUS] & TDRE);
    }


/**********************************
*                                 *
* here are all the parallel drivers *
*                                 *
**********************************/
```

```
/*
 * initialize a pia
 */
init_6821(port,wbits,setval)
    char port[], wbits, setval;
    {
    port[PCONTROL] = 0;
    port[PDIRECTION] = wbits;
    port[PCONTROL] = setval;
    port[PDATA] = '\0';
    }


/*
 * put a character to the printer
 * expanding newline with (lf)(cr)
 */
parout(c,port)
    char c, port[];
    {
    if (c == '\r')
        poutd(LF,port);
    poutd(c,port);
    return(c);
    }
```

```
/*
 * Poll the printer, and if it's ready,
 * then, clear the ready flag and send
 * the character.
 */
poutd(c,port)
    char c, port[];
    {
    char junk;

    while (parost(port) == NOTREADY);
    junk = port[PDATA];
    port[PDATA] = c;
    return(c);
    }


/*
 * check the pia irqbit
 * status
 */
parost(port)
    char port[];
    {
    return (port[PCONTROL] & PRDY_BIT);
    }
```

OS9 USER NOTES

By: Peter Dibble
517 Goler House
Rochester, NY 14620

More on Locking

Last month I discussed shared data modules, and demonstrated a locking method which could be used to permit only one process at a time to access a data module, or, for that matter, any shareable resource.

The locking protocol I demonstrated last month has two serious problems. One is only a problem for those who, like most of us, can only run more than one process by sharing a processor between several processes. The other problem limits the usefulness of concurrent processes. Both problems have solutions.

The locking algorithm I demonstrated last month used a technique called "busy waiting." This is usually the easiest way to make a process wait until something happens, but it wastes processor cycles. I tried to reduce the amount of time wasted in the locking module as much as possible by putting a "sleep" in its wait loop, but the solution I gave was, nevertheless, inefficient. If waiting for the lock uses processor time, even very slowly, all you have to do is line up enough processes waiting for the lock and you can slow the computer down to a crawl. You might think that you could always make the waiting processes as cheap to run as necessary by putting a longer sleep into the wait loop, but if the sleep is made very long, there may be a significant time during which the lock is off and the all the processes which want it are sleeping. If the goal is performance, (using performance in the same sense as "high performance car") it is not good to leave a scarce resource like the lock unused for any length of time. The goal is to design an algorithm which allows waiting processes to be completely idle until the lock is available, then awakens one process and gives it the lock.

If each process is running on its own processor, the processor running a waiting process has nothing better to do than zip around the wait loop. Some people think busy waiting is bad even then. I tend toward the opposite extreme. The problems with busy waiting are obvious, the alternatives have trickier problems. The issues involved in choosing a busy waiting algorithm over a more sophisticated one are much like those involved in choosing a bubble sort over one of the flashier sorting algorithms, that is, for a small problem the simple algorithm will do fine.

The other problem with the locking algorithm I gave is that it permits "lockout," ie. a process can wait forever without ever getting the lock even when no other process holds the lock forever. If there will seldom be a process waiting for the lock, lockout isn't a big problem, but for locks that usually have a process or more waiting for them lockout is an important consideration.

It is tricky to detect lockout in an algorithm, but here are the basic rules for finding it: Imagine that you are controlling the computer's dispatcher (deciding which process runs and for how long). It is your job to prevent a certain process from ever getting the lock. You may run any mix of programs you like any way you like except that the process you are trying to prevent from getting the lock must be allowed to run every now and then. If it is possible to prevent that process from ever getting the lock, there is lockout. The sequence of events that demonstrates that lockout is possible for the algorithm I gave last month is: Two processes are running, A and B. Both processes are simple programs which just get the lock then release it again and again. Either process could be locked out, but the following "execution sequence" only demonstrates that process B can be locked out.

```
Start
  A: lock
  B: try to lock
  A: unlock
  A: lock
  B: try to lock
  A: lock
  etc.
```

You see that by allowing process A to run long enough so that can get the lock again each time it releases it I can shut process B out completely. This may seem unfair, but it shows that the algorithm permits lockout. Murphy's law certainly dictates that if it is possible to prevent a process from ever getting the lock (and you want it to get the lock), the improbable execution sequence which leads to lockout will happen at the worst possible moment. This is one of the kinds of problem that cause strange behavior in complicated systems.

There are many ways to do locking that don't use busy waiting or have deadlock. I am not going to discuss these tricks this month, but I will leave you with two hints. The OS-9 SEND service request offers an alternative to busy waiting, locking can be done without deadlock by using any of several algorithms including one called the Doorman Algorithm.

Getting a Good "Mix"

The standard use for multiple processes is to make maximum use of a processor when the work to be done involves a lot of waiting for outside events, such as terminal input. A process could spend most of its time waiting for input from a terminal, and delegate any major work to child processes. This way the program would almost always be ready to accept input from the terminal, even when some previous piece of work was still in progress. Using a special process to print a screen is a particularly apt use of this principle. There is really no reason why someone should have to wait for a print request to complete before continuing,and there is usually no need for the process that is doing the printing to communicate with its parent process. The process that is driving the printer spends most of its time waiting for the printer, and the process that is responsible for the screen is, very likely, spending most of its time waiting for input from the terminal. These tasks can be in progress at the same time with almost no effect on one another. When one process is waiting for something, the other process can run without interference. With tasks like printing and screen handling, the computer will spend most of its time with both processes waiting.

Some programs run well together, other programs interfere badly with each other. Finding good sets of programs to run at the same time, and adjusting their priorities so they all will run as fast as possible is called finding a good "mix." Tuning hardware and software so a single program can run as fast as possible is a complicated job, but choosing groups of programs which will run well together, and tuning the system so the groups will run as fast as possible is more of a black art. I like to keep my personal computer rather lightly loaded (no more than two or three processes active at a time), but it is good question just how much time a computer should spend waiting. If you give the machine so much work to do that it never has to wait, each process will run slowly. A computer that has no resources in reserve is said to be saturated.

Consider the case of a program which is reading from the terminal. Usually, in a saturated computer, there are several processes waiting for processor time at any moment. The process waiting for the input character will have to wait for at least one process, maybe several, to have their turn before it will get a chance to run. If each process gets a turn one tenth of second long, and there are an average of two processes waiting to run, then a process will take about two tenths of a second to respond to a simple keystroke. That comes to 5 characters per second, or 300 characters per minute. For perspective, my terminal repeats at 10 characters per second.

Fortunately, programs running under OS-9 don't actually do any I/O. OS-9 is arranged so that input and output are done by OS-9 rather than by user programs. The device drivers are responsible for all I/O. OS-9 always gives very fast service to device drivers. Almost anything will be interrupted to allow a device driver to deal with input or output. Some device drivers have a reservoir for 100 (or so) characters which they can save up and give to a process in a burst next time the process is started.

For the best performance a computer should be kept idle most of the time, that way it will immediately jump on any work you give it. It would be nice to have enough money to buy overpowered computers so there would be lots of idle time and excellent performance, but if money is a concern you have to strike a compromise between getting fast response, and getting the maximum amount of work out of your machine.

It is possible to speed up important processes by changing their priority. The heavier the load on a computer, the more important it is to fuss with priorities. An edit session, a listing to the printer, and an assembly can share the machine very nicely if the priorities are properly set. The edit session is interacting with an impatient human, so it should have a high priority assigned to it. Since editing usually involves a lot of dead time while the human doing the editing stares at the screen, the editor will actually use very little processor time. The process that is printing is very much the same story. It isn't interacting with a human, but even a 200 character per second printer is slow by computer standards. The process that is driving the printer should be given an intermediate priority so it will be able to run the printer at a good clip without interfering to any great extent with the edit process. The assembly should be given a very low priority. Assemblies are the type of thing that will use a lot of processor time if they are allowed. Even if it is given a low priority, the assembly will get time that the other processes don't want, so since both will usually be waiting for something, the assembly will get plenty of time.

Most business programs, as well as compilers, assemblers, and disk utility programs, spend a lot of time waiting for the disk to do something. The sound of a disk clucking and buzzing is a pleasant busy sound, but it actually signifies wasted time. While the disk is doing mechanical things like starting, seeking, loading the head, and even turning, some program is likely to be waiting. OS-9 makes some effort to speed disk access, but with several processes wanting to access the same disk the problem is more than a small operating system can handle. There are standard tricks for reducing the amount of time a program spends waiting for the disk drive. The easiest of these for a regular user to get at is the use of large buffers. Most programs that access the disk will run faster if they are given enough storage so they can read and write large blocks of data. If you want to hear some very busy noises from your drives, start a COPY with only a little bit of memory, then do a DIR for a large directory on the same disk you are COPYing on. The disk drive will chuckle madly as it shuttles back and forth from directory to file in an attempt to serve both the copy program and the DIR command. Switching from file to file on a disk (even a Winchester) is slow. The best way to deal with this is to avoid the problem by not running more than one program accessing a particular drive at a time. It will be obvious if there is a problem. If programs are run in the wrong combinations, they will run very slowly, and the disk will sound very active. If you have to make the best of a bad mix, give processes as much memory as you can. Well designed programs can use extra storage to cut down disk usage, or to transfer (read or write) more data for each turn they get.

An Assembly Language Program Which Sets Printer Options

I just installed a new printer on my system, an Okidata Microline 92 (nice printer). I used to set the options on my MX80 with a group of procedure files. An example would be the file called Comprint which contained the command "display 0f >/p". It would have been possible to set the printer to compressed printing mode by typing the display command instead of invoking the procedure file by typing /d0/comprint, but I can never remember the Epson control codes. Installing a new printer seemed like a good excuse to find a better way of setting the printer options. The program POpt is the first complete assembly language program I have published here. I hope you find it as useful as I do.

POpt doesn't do anything technically exciting, but it is a fairly simple assembly language program which includes most of the elements found in assembler programs. I am going to go through the interesting points of the program moving generally from the beginning to the end.

The NAM and TTL statements in the first two lines of the program are purely cosmetic. They provide information which the assembler puts in the page headings. The block following those two lines is the introductory comment for the program. All comments might be considered cosmetic, but although they don't generate any code, I think of them as essential parts of assembly language programs. Any line with an asterisk in column one is a comment; the box I draw around the comment is just to make it look nice.

If you are looking at the output of the assembler, the two lines after the introductory comment are IFP1 and ENOC. In the original source there is a line between these, "use /d0/defs/defsilist", which calls in a list of USE commands which make files containing all the system definitions part of the program. These definitions help make the rest of the program more readable. The words Prgrm+Objct would have to be replaced with the much less understandable $11 if the system definitions, or some other similar set of definitions, weren't included in the program. Throughout the rest of the program I used the symbols defined in the definitions files (and a few additional SET commands in the program itself) whenever I could. The IFP1/ENDC which is wrapped around the use statement prevents the extra files from being read on the second pass the assembler takes through the file. No statement in the system definitions defines any memory so there is no reason for the assembler to read it on both the first and second passes; not reading it during the second pass saves a good deal of time, and prevents the lines in the definitions from being included in the program's line numbering. The lines used from the definitions

files don't print both because they aren't read on the second pass (when output is generated), and because the first line in the definitions file is the assembly directive OPT -l which directs the assembler not to print anything until it encounters the OPT l directive. The definitions files I routinely include in assemblies are:

OS9Defs
OS9SysDefs
OS910Defs
OS9R8FDefs
OS9SCFDefs

There are a lot of symbols in all those files; a program with the full set of definition files generally needs to be assembled in a region of at least 24K to accommodate the large symbol table. If you want to use your memory more economically, create a stripped down definitions file with only the definitions you expect to use, and use it instead of the standard files; but be prepared to scrap your file and build a new one if you get a new version of the operating system. Level One and Level Two definitely have different definitions, and if you dig around deep enough in the operating system, it is unwise to count on things staying fixed even from version to version.

A few lines down from the ENDC is the MOD statement. This statement generates the OS-9 module header, a block of data which OS-9 needs. The fields in the module header are:

Program length -- Trying to fill in a number here would be foolish. The assembler can figure out the length of the module for you. The symbol PgmLen is defined in the last line of this program.

Symbol used for program name location -- This isn't the program name itself, but the name you choose to assign to the location containing the name. I like the name "Name" for that location. The program's name is usually placed close to the module header, but it can be placed elsewhere in the module if it is convenient for you to do it that way.

Module Type -- I like to define the module type as a symbol before the MOD statement and just put the symbol here. The module type tells OS-9 what kind of thing to expect this module to do. This module is a program (not a subroutine or data), and it consists of object code (not data or some sort of intermediate code).

Revision -- This field contains two types of information. It indicates whether the module is reentrant, usable by several different users at the same time, and gives the revision number of the program. Most well written programs are reentrant, so, since OS-9 uses reentrant modules more efficiently than non-reentrant modules, most programs should be labeled reentrant. The revision number is used when a module is loaded from disk to determine whether the module should replace a module by the same name already in memory. A module with a higher revision number will replace a module with a lower number.This is particularly useful if you want to override a module which has been placed in ROM. Unless you want to supersede a module in memory the revision should be 1.

Entry point -- The name assigned to the first instruction in the program. I usually insert a line before the first instruction in the program with this name on it. This saves a little bit of typing if I want to add instructions before the first instruction in a program.

Minimum amount of permanent storage required -- The amount of storage the program will need in addition to the storage used for the module itself. This number, like the program length, can be calculated by the assembler -- note the MemSize equ * a few lines down. The line after the MOD statement tells the assembler to reserve space for one byte of storage. The next two lines reserve 255 more bytes. The total memory requirements of this program are one byte for the printer's path number, and 255 bytes for the stack. The stack probably doesn't need to be that large for this simple program, but OS-9 is going to allocate memory in 256 byte units even on a level I system, so I played it safe and squandered the memory on the stack. The results of allocating too little space for the stack are very unpleasant.

The equate after the rmb for the stack uses the "." special symbol which means the current offset in the data definitions. This is an easy way to get the assembler to tell us how much storage the program will need for use in the MOD statement.

The next two lines are the module's name (pointed to by the module name field in the module header), and the version number. The module name must be defined with a FCS statement. This type of data definition closes the string it is defining by setting the high order bit on in the last byte -- "t" is $F4 instead of $74. This lets OS-9 know where the end of the name is. The byte after the name is by convention the version number of the program. Some utility programs display this number, but it is optional. Nothing awful will happen if you start right in with data or program after the program name.

The version number is the last overhead until the very end of the program. The fcc's and fcb's for the next 40 or so lines define constants needed in the program. About the only interesting thing about them is that each of the strings defined with a fcc is followed by fcb C$CR, a carriage return. At first it looks like I could have saved space by using one <CR> for all the strings, but it turns out that the extra code needed for that approach uses more memory than the extra carriage returns.

The program scans the parameter string, and if certain characters are found, sends character strings to the printer. There are three phases: first the input length, in D, is checked. If it is one (or lower) there is no parameter string; in this case display a menu of options. Second, scan the parameter string for the character "/" which denotes a device name. If there is a device name in the parameter, open that device as the printer, otherwise open the device /P. Third, scan the parameter string again ignoring any characters in a device name. Translate each character to upper case and compare the translated character to each significant character. Each time a significant character is found, transmit the appropriate character string to the printer, and send a line to the standard output path describing what has been done.

There are a couple of simple tricks which are useful while scanning the parameter string. The shell always terminates the parameter string with a carriage return. This lets me terminate the scan when I encounter a carriage return instead of having to count bytes. Data bytes may have the parity bit on or off. I remove the parity bit with "anda $7F." If the parity bit is left on, twice as many comparisons need to be done. For example, "a" could be $61 or $E1. In this case, I thought it would be best to treat both upper and lower case characters as the same. The easiest way to do this is to translate all lower case letters to upper case (or vice versa if you like). Once you determine that a character is an upper case letter it can be translated to a lower case letter by subtracting $20 from it, or anding $11011111 with it.

There are two sections of this program responsible for output. Common1 writes strings two bytes long to the printer. It uses the I$Write service request which writes a specified number of characters without any editing. There is nothing special about two bytes; it is just the length of the longest control string I wanted to be able to send to the printer. I padded the shorter control strings to two bytes by adding a $00, a null, to them. Common2 writes up to 80 characters to the standard output path. Common2 uses the I$WritLn service request which treats the carriage return as a special case. When it encounters a carriage return it does whatever the path descriptor is set up to do on end of line (normally send <CR><LF>) and returns. This means that by terminating each string to be written by Common2 with a <CR> I make it unnecessary to know the length of any of the strings.

This program ends in either of two places. If there are no errors, after the second scan the program branches to Exit which clears the carry bit in the condition code and performs the F$Exit service request returning control to OS-9. If there is an error, control goes to ErrXit which sets the carry bit and returns control to OS-9. You might expect that the best way to set or clear the carry bit in the condition code register is with the andcc and orcc instructions. Those instructions certainly are able to turn the carry bit on and off, but the COM instruction turns the carry bit on faster (and the CLR instruction turns it off faster) than the obvious instruction. Whenever the A or B accumulator is free, it is fastest to set or clear the carry flag by playing with the accumulator.

At the very end of POpt there are two final lines of overhead. The EMOD directive causes the assembler to generate a checksum for the module which is used when this program is run to make certain that the module is valid and undamaged. The line with "PgmLen equ *" calculates the length of the module for use in the MOD statement at the very beginning of the module.

The OS-9 User's Group

An OS-9 User's Group was formed last summer. I couldn't say it's thriving, but it is coming along. The club has a telephone bulletin board, and lots of dreams. It isn't going to go anywhere unless plenty of OS-9 users join it. Membership is $25 for individuals (payable to OS-9 Users Group c/o Terry Straehley 1005 Roble Lane, Santa Barbara CA 93103). I strongly suggest that all OS-9 users join the group. Even with the relatively small membership the group now has, a lot of interesting information passes through the bulletin board. If we all join, this group could become a great resource.

The Future of this Column

There is enough material for another six months or more of columns about concurrent processes, but I am going to move on to some other subjects for a while. It seems there are a great many new OS-9 users out there, some of whom have written to me asking for help with the fundamentals of the system. The program this month is a first attempt to help these people. I'll try to devote at least part of this column to OS-9 basics for the next few months.

```
POpt - Change Printer Setup Options for ML 93

00001                          NAM  POpt
00002                          TTL  Change Printer Setup Options for ML 93
00003    0----------*-------------------------------*-----------------0
00004    *  Printer Setup Options                                      *
00005    *    c     Correspondence Quality                            *
00006    *                                                            *
00007    *    0     Ten CPI                                           *
00008    *    2     12 CPI                                            *
00009    *    7     17 CPI                                            *
00010    *          Double Width Characters                           *
00011    *    5     Five CPI                                          *
00012    *    6     Six CPI                                           *
00013    *    8     Eight CPI                                         *
00014    *                                                            *
00015    *    r     reset to initial conditions                       *
00016    *                                                            *
00017    *    /     Lead in for alternate path name. Default          *
00018    *          is /P. The path name must either be the           *
00019    *          last parameter, or separated from the next        *
00020    *          parameter by a delimiter.                         *
00021    *                                                            *
00022    *  The options are specified as parameters when              *
00023    *  POpt is run. If no options are specified, a menu          *
00024    *  is presented.                                             *
00025    *                                                            *
00026    *  Examples:                                                 *
00027    *       popt rc2                                             *
00028    *  -)     Printer Reset                                      *
00029    *  -)     Correspondence Quality Printing                    *
00030    *  -)     Print Density twelve characters per inch           *
00031    *                                                            *
00032    *       popt r6 /p1                                          *
00033    *  -)     Printer Reset                                      *
00034    *  -)     Print Density six characters per inch              *
00035    *  -)   (output was directed to the printer at /p1 )e        *
00036    *  You can put the print options on either or both           *
00037    *  sides of the device name...                               *
00038    *  popt rc /p1    is the same as    popt r /p1 c             *
00039    0------------------------------------------------------------0
00040                          IFP1
00042                          ENBC
00043  0011            Type    set   Prgrm+Object
00044  0081            Revs    set   ReEnt+1
00045  0000 87CD0A0A           MOD   PgmLen,Name,Type,Revs,Entry,MemSize
00046 B 0000            PrintBf rmb   1
00047  00FF            StackSz set   255
00048 B 0001                    rmb   StackSz    space for stack
00049 D 0100            Memsize equ   .
00050  0000 504F704F4  Name    fcs   /POpt/
00051  0011 01         Edition fcb   1
00052  0001            StdOut  set   1          Number of Standard Output Path
00053  0012 2F50       DPrtNam fcc   "/P"       Default Printer Name
00054  0014 00                 fcb   C$CR
00055        ********
00056        * Responses for each printer option set
00057        *
00058  0015 5072696E   Msg5CPI fcc   /Print density five characters per inch i
00059  004A 00                 fcb   C$CR
00060  004B 5072696E   Msg6CPI fcc   /Print density six characters per inch is
00061  007F 00                 fcb   C$CR

Microware OS-9 Assembler 2.1  06/11/83 16:51:56                    Page 002
POpt - Change Printer Setup Options for ML 93

00062  0080 5072696E   Msg8CPI fcc   /Print density 8.5 characters per inch is
00063  00B4 00                 fcb   C$CR
00064  00B5 5072696E   Msg10CPI fcc  /Print density 10 characters per inch is
00065  00E2 00                 fcb   C$CR
00066  00E3 5072696E   Msg12CPI fcc  /Print density twelve characters per inch
00067  0118 00                 fcb   C$CR
00068  0119 5072696E   Msg17CPI fcc  /Print density 17 characters per inch/
00069  0138 00                 fcb   C$CR
00070  0139 436F7272   MsgC0    fcc  /Correspondence Quality Printing/
00071  0158 00                 fcb   C$CR
00072  0159 5072696E   MsgRst   fcc  /Printer Reset/
00073  0172 00                 fcb   C$CR
00074        ******
00075        * Printer Control Strings
00076        *
00077  0173 1B31       CCQS    fcb   $1b,'1    Set correspondence quality
00078  0175 1E1F       CC5CPI  fcb   $1E,$1F   Five CPI
00079  0177 1C1F       CC6CPI  fcb   $1C,$1F   Six CPI
00080  0179 1D1F       CC8CPI  fcb   $1D,$1F   Eight CPI
00081  017B 1E00       CC0CPI  fcb   $1E,0     Ten CPI
00082  017D 1C00       CC2CPI  fcb   $1C,0     Twelve CPI
00083  017F 1D00       CC7CPI  fcb   $1D,0     Seventeen CPI
00084  0181 1B00       CCRst   fcb   $1B,0     reset printer
00085        **********
00086        * The Menu
00087        *
00088  0183 4E6F2060   ErrMsg1 fcc   /No more than 127 bytes of parameters are
00089  01A0 00                 fcb   C$CR
00090  01A1 492F4F20   ErrMsg2 fcc   .I/O error on printer path.
00091  01BA 09                 fcb   C$CR
00092  01BB 506F7074   Menu1   fcc   /POpt accepts the following parameters:/
00093  01E1 00                 fcb   C$CR
00094  01E2 20202020   Menu2   fcc   / R - Reset the printer/
00095  01FB 00                 fcb   C$CR
00096  01FC 20432020   Menu3   fcc   / C - Correspondence quality print/
00097  021A 00                 fcb   C$CR
00098  021B 20532020   Menu4   fcc   / S - Print at five characters per inch/
00099  0241 00                 fcb   C$CR
```

```
00100  0282 20342028  Menu5    fcc   / 5 - Print 40 six characters per inch/
00101  0287 0D                  fcb   C#CR
00102  0288 20382028  Menu8    fcc   / 8 - Print at eight and a half character
00103  028A 0D                  fcb   C#CR
00104  028B 20322028  Menu7    fcc   / 0 - Print at ten characters per inch/
00105  028D 0D                  fcb   C#CR
00106  028E 20312028  Menu0    fcc   / 2 - Print at twelve characters per inch
00107  0290 0D                  fcb   C#CR
00108  0291 20372028  Menu9    fcc   / 7 - Print at eleven characters per i
00109  0295 00                  fcb   C#CR
00110  0296           Entry
00111       ****************
00112       * U points to the start of the parameter area.
00113       * Y points to the end of the parameter area.
00114       * The last character in the parameter area is a <CR>.
00115       * D contains the length of the parameter area.
00116       *
00117  0296 10830001            cmpd  #1          Check length of parameter area
00118  029A 10230137            lbls  Menu        If there is nothing there? Dis
00119  029E 10830080            cmpd  #128        It's hard to deal with paramet
00120  02A2 1024017E            lbhs  Error1      Input parameter area too long

00121       *-------------------------------------------*
00122       * Search parameters for output device override*
00123       *-------------------------------------------*
00124  02A6 3436                pshs  D,X,Y       save everything
00125  02A8           Loop1
00126  02A8 A680                lda   ,X+
00127  02AA 847F                anda  #$7F        clear parity bit
00128  02AC 810D                cmpa  #$0D        <CR>?
00129  02AE 2713                beq   Loop1B
00130  02B0 812F                cmpa  #'/         start of path name?
00131  02B2 26F4                bne   Loop1
00132  02B4 301F                leax  -1,X        back up one to /
00133  02B6           Loop11
00134       *-------------------------------------------*
00135       *     Open alternate printer path           *
00136       *-------------------------------------------*
00137  02B6 8602                lda   #Write.
00138  02B8 103F84              OS9   I$Open
00139  02BB 10250134F           lbcs  Error2
00140  02BF 9700                sta   PrtPth#     save the path number
00141  02C1 2008                bra   Loop1E
00142       *-------------------------------------------*
00143       *  No alternate printer path found          *
00144       *-------------------------------------------*
00145  02C3           Loop1B
00146  02C3 308D6CCB            leax  DPrtNam,PCR
00147  02C7 20EB                bra   Loop11      Open the default printer path
00148  02C9           Loop1E
00149  02C9 3536                puls  D,X,Y       restore

00150       *-------------------------------------------*
00151       * Loop2 scans the parameter string for      *
00152       * printer control options. If an option is  *
00153       * found the corresponding subroutine is     *
00154       * called.                                   *
00155       *-------------------------------------------*
00156  02CB           Loop2
00157  02CB A680                lda   ,X+
00158  02CD 847F                anda  #$7F        clear parity bit
00159  02CF 810D                cmpa  #$0D        <CR>?
00160  02D1 2738                beq   Exit
00161  02D3 8120                cmpa  #$20        control character?
00162  02D5 23F4                bls   Loop2       yes! loop
00163  02D7 812F                cmpa  #'/         start of a path name?
00164  02D9 10270DE0            lbeq  SkipPN      Yes; Skip over the path name
00165       *-------------------------------------------*
00166       *     Translate lower to upper case if      *
00167       *     necessary.                            *
00168       *-------------------------------------------*
00169  02DD 8161                cmpa  #'a
00170  02DF 2506                blo   Loop21
00171  02E1 817A                cmpa  #'z
00172  02E3 2202                bhi   Loop21
00173  02E5 8020                suba  #$20        lower to upper case
00174  02E7           Loop21
00175       *-------------------------------------------*
00176       *        Analyse the parameter              *
00177       *-------------------------------------------*
00178  02E7 8152                cmpa  #'R         reset?
00179  02E9 2734                beq   Reset
00180  02EB 8143                cmpa  #'C         Correspondence quality?
00181  02ED 2739                beq   CQ
00182  02EF 8130                cmpa  #'0         Ten CPI
00183  02F1 2743                beq   TenCPI
00184  02F3 8132                cmpa  #'2         Twelve CPI
00185  02F5 2751                beq   TwlvCPI
00186  02F7 8135                cmpa  #'5         Five CPI?
00187  02F9 2760                beq   FiveCPI
00188  02FB 8136                cmpa  #'6         6 CPI
00189  02FD 2766                beq   SixCPI
00190  02FF 8137                cmpa  #'7         Seventeen CPI
00191  0301 276E                beq   SvnteCPI

00192  0303 8138                cmpa  #'8         Eight and a half CPI?
00193  0305 10270088            lbeq  EightCPI
00194  0309 20C0                bra   Loop2
00195  030B           Exit
00196  030B 5F                  clrb              set B (return code) to 0 and t
00197  030C 103F06              OS9   F$Exit      return to OS-9

00198  030F           Reset
00199  030F 3410                pshs  X
00200  0311 308DFB98            leax  CCRst,PCR   point at Reset control string
00201  0315 17008F              lbsr  Common1     write it
00202  0318 308DFB35            leax  MsgRst,PCR  point at remark
00203  031C 170094              lbsr  Common2     write it
00204  031F 7510                puls  X
00205  0321 20A8                bra   Loop2       go search for next option
00206  0323           CQ
00207  0323 3410                pshs  X
00208  0325 308DFB84            leax  CCCQ,PCR
00209  0329 6D7C                tst   Common1
00210  032B 308DFBE2            leax  MsgCQ,PCR
00211  032F 170081              lbsr  Common2
00212  0332 7510                puls  X
00213  0334 2095                bra   Loop2
00214  0336           TenCPI
00215  0336 3410                pshs  X
00216  0338 308DFDAB            leax  CC0CPI,PCR
00217  033C 0D19                tst   Common1
00218  033E 308DFCF3            leax  Msg10CPI,PCR
00219  0342 0D4F                tst   Common2
00220  0344 3510                puls  X
00221  0346 2083                bra   Loop2
00222  0348           TwlvCPI
00223  0348 3410                pshs  X
00224  034A 308DFD98            leax  CC2CPI,PCR
00225  034E 0D57                tst   Common1
00226  0350 308DFD6F            leax  Msg12CPI,PCR
00227  0354 0D58                tst   Common2
00228  0356 3510                puls  X
00229  0358 16FF70              lbra  Loop2
00230  035B           FiveCPI
00231  035B 3410                pshs  X
00232  035D 308DFD80            leax  CC5CPI,PCR
00233  0361 0D44                tst   Common1
00234  0363 308DFC2E            leax  Msg5CPI,PCR
00235  0367 0D4A                tst   Common2
00236  0369 3510                puls  X
00237  036B 16FF50              lbra  Loop2
00238  036E           SixCPI
00239  036E 3410                pshs  X
00240  0370 308DFD6F            leax  CC6CPI,PCR
00241  0374 0D31                tst   Common1
00242  0376 308DFC51            leax  Msg6CPI,PCR
00243  037A 0D37                tst   Common2
00244  037C 3510                puls  X
00245  037E 16FF40              lbra  Loop2
00246  0401           SvnteCPI
00247  0401 3410                pshs  X
00248  0403 308DFD64            leax  CC7CPI,PCR
00249  0407 0D1E                tst   Common1
00250  0409 308DFCFF            leax  Msg17CPI,PCR
00251  040D 0D24                tst   Common2
00252  040F 3510                puls  X
00253  0411 16FF37              lbra  Loop2
00254  0414           EightCPI
00255  0414 3410                pshs  X
00256  0416 308D7D40            leax  CC8CPI,PCR
00257  041A 0D0D                tst   Common1
00258  041C 308DFC60            leax  Msg8CPI,PCR
00259  0420 0D11                tst   Common2
00260  0422 3510                puls  X
00261  0424 16FF24              lbra  Loop2

00262  0427           Common1
00263  0427 9600                lda   Prt Pth#    Printer Path Number
00264  0429 108E0002            ldy   #2          length
00265  042D 103F8A              OS9   I$Write
00266  0430 2505                bcs   Error2      I/O error on printer path
00267  0432 39                  rts
00268  0433           Common2
00269  0433 8601                lda   #StdOut     output path for remarks
00270  0435 108E0050            ldy   #80         max length of strings
00271  0439 103F8C              OS9   I$WritLn
00272  043C 39                  rts
00273       *-------------------------------------------*
00274       *   Skip over alternate printer path name   *
00275       *-------------------------------------------*
00276  043D           SkipPN
00277  043D A680                lda   ,X+
00278  043F 847F                anda  #$7F
00279  0441 810D                cmpa  #C$CR       <CR>?
00280  0443 1027FF44            lbeq  Exit        yes! done
00281  0447 8120                cmpa  #C$SPAC     (space)?
00282  0449 1027FEFE            lbeq  Loop2       end of path name
00283  044D 812C                cmpa  #',
00284  044F 1027FEF8            lbeq  Loop2       end of path name
00285  0453 20EB                bra   SkipPN
```

```
00286  0435          Mpnu
00287  0435 308D6B42          leax  Mpnu1,PCR
00288  0459 0000              bsr   Common2
00289  045B 308D6B4D          leax  Mpnu2,PCR
00290  045F 0002              bsr   Common2
00291  0461 308D6B64          leax  Mpnu3,PCR
00292  0465 0BCC              bsr   Common2
00293  0467 308D6D00          leax  Mpnu4,PCR
00294  046B 00Cb              bsr   Common2
00295  046D 308D6D01          leax  Mpnu5,PCR
00296  0471 0900              bsr   Common2
00297  0473 308D6D01          leax  Mpnu6,PCR
00298  0477 0A0A              bsr   Common2
00299  0479 308D6E1E          leax  Mpnu7,PCR
00300  047D 0904              bsr   Common2
00301  047F 308D6E3E          leax  MpnuB,PCR
00302  0481 05AE              bsr   Common2
00303  0485 308D6E61          leax  MpnuY,PCR
00304  0489 05A0              bsr   Common2
00305  048B 1AFE7D            lbra  Exit
00306         *-------------------------------------*
00307         *          End with an Error          *
00308         *-------------------------------------*
00309  048E          Error2   equ   *         Error in printer path
00310  048E 3404              pshs  B         save error code
00311  0490 10BE0050          ldy   #80
00312  0494 308D6B09          leax  ErrMsg2,PCR
00313  0498 8602              lda   #2
00314  049A 103F8C            OS9   I$WritLn
00315  049D 3504              puls  B         recover error code
00316  049F 103F8F            OS9   F$PErr    print error message
00317  04A2 2006              bra   ErrExit

00318  04A4          Error1   equ   *         Parameter string too long
00319  04A4 10BE0050          ldy   #80
00320  04A8 308D6CC3          leax  ErrMsg1,PCR
00321  04AC B602              lda   #2        Error output
00322  04AE 103F8C            OS9   I$WritLn
00323  04B1 CA01              orb   #1        error code
00324  04B3          ErrExit
00325  04B3 43                coma           set carry
00326  04B4 103F06            OS9   F$Exit
00327  04B7 2FD039            LBEQ
00328  04BA          PgmLen   equ   *
```

MACROBUFFER ...REVIEW

John P. Tucker
12015 O'Connor Road, #184
San Antonio, Texas 78233
512-653-9834

A Review of Macrobuffer from ChiraTech Scientific Instruments; 204 North Link Lane, Alpha 9; Fort Collins, Colorado 80524

OK, so you don't have to believe me, but I have a 6800 system up and running with 176K of memory in it. I can run my Epson MX-100 for hours of constant printing and it never once makes a demand on the CPU. In fact, I can run two printers at the same time on a system that does NOT use interrupts. One printer runs entirely free of the other. While my Epson is busy with a long, long printout, my computer is free for me to use however I may wish. There being no interrupts, I am almost unaware of the task the printer is performing. The only thing that keeps me alert to it is its constant printing while I am tarrying with entries on the keyboard of the terminal.

What's going on? I'm using one of ChiraTech Scientific Instruments' new Macrobuffer cards for the SS-30 bus.

The Macrobuffer is an intelligent printer card and much more. It is plugged into the SS-30 bus (remember that the SS-30 bus is the output bus for your SS-50/50C computer) and appears to the computer to be an extremely high speed printer. In fact, it appears to be a printer that "soaks up" output at the rate of about 447.4 kHz (parallel output version) or about 2800 characters per second in the serial version. Output to the printer is at

protocol rate; parallel at whatever speed the printer is handshaking from the computer, serial at whatever baud rate was selected. (Output of the serial version can be user selected from 110 baud to 9600 baud.)

Using single side, single density 5 1/4 inch disks, as I do, one can dump about one and a half disks of processed text into the buffer before it becomes filled. This is a bunch of printing, and the disks shut down and do nothing until the initial read has been output. The depth to which the large buffer is drained before the next disk read is required is programmable from 1K to 127K. To quote the excellent documentation, "With 131,072 bytes of high-speed buffer memory, the Macrobuffer can hold over 32 FULL (4000-character) pages of print. (Even more if the pages aren't full.)" That's impressive and it's functional. It allows the computer to be free for any other task chosen while a printout occurs; the printout could be the results of a long Basic (or other language) program, or it could be the output from processed text. OR! It could be a mixture of both since you will seldom fill that big a buffer. You can perform multiple tasks sequentially while the printer does its thing at its own speed and you do your thing at yours.

Furthermore, the Macrobuffer is intelligent. Very intelligent. In fact, there's a 6809 on board the card. There are 20 text formatting, character filtering, and user-oriented control features that are invoked by software sequences.

These 20 features range from such things as setting margins (top and/or left), page length and size, tabs, etc. One can program the card for the printer to ignore any given character, translate an input character to another, and do many more useful "tricks." I find the character translation facility extremely useful. In my application for text processing, many of my files include the reverse apostrophe (`) which is mistranslated by many daisy wheel printers and some other matrix printers. To correct this, one issues a simple command to the card and, in this example, causes the reverse apostrophe to be printed as the standard apostrophe. Any character substitution can be inserted as desired.

In the limited space available here it is impossible to review this card in full. Several pages of '68' Micro Journal would be required simply to outline the card's capabilities. It's usefulness can only be experienced and not explained.

For a while I was a bit puzzled by one requirement of the card. It looks like a SERIAL device to the computer, but it is a parallel output card (if ordered in such a configuration.) The serial print driver routine is ALWAYS used with the card, regardless of the printer attached, serial or parallel. This is no problem, but this protocol must be observed.

(A word of caution at this point: If you are using one of the older Basic languages that uses the PORT= instruction, you will have to place this card in a port slot where the Basic will look for a SERIAL device; usually Port 1 or 2 or 3. The old SWTPCo Basic, for example, looks for an MP-C card on Port 0, and parallel cards on ports 4 through 7, usually with the disk controller on ports 5 and 6. Serial cards were "permitted" only on ports 1 through 3.)

While my system, used for reviewing this card, is a 6800 SWTPCo system, don't get the impression that this card is restricted to this type of unit.

ANY COMPUTER USING THE SS-30 BUS CAN USE THIS CARD! And I endorse your investigation of using it. For FLEX9 users, a Printer Control package is available to allow sequences of commands to be sent to the card for activating various functions. For users such as myself, old dinosaurs with FLEX2 and 6800's, very simple programs can be written and stored on disk for future use. I have had NO training in machine language routines and yet I was able to create several useful files using only LDAA and JSR with a final JMP WARMS. (FLEX is wonderful and user friendly, isn't it?)

For you hardware freaks, the card contains 64K of 4164 Dynamic RAM, a 6809 processor, a PIA (on the parallel output version), an ACIA (two on the serial version), a 2716 EPROM, a 2016 scratchpad RAM, and a handful of support chips, some of which vary according to which version of the card is ordered. Many of the chips are 2 mHz devices, indicating the thoughtfulness of the designer. The card itself is very well done. In fact, the card is so well done that I suspect it was designed by computer aid. I found no cut traces or added jumpers that are too common on first runs of intricate (and some not so intricate) cards. The several programming jumpers are plug-on types so that no soldering is required to configure the card to your system. I had no configuring to do; the factory defaults fit my system perfectly.

The price will be in the $450 bracket, running slightly under for some configurations and slightly over for others, but not by much. Considering that you are buying 128K of memory, this price is almost unbelievably low. I remember paying approximately that much for the first 8K of memory to go in my system.

Not enough can be said of the excellence of the documentation for this board. I have never, without exception, encountered a better or more comprehensive or more comprehensible set. Believe it or not, I had no unanswered questions after reading the 29 pages of lucid instructions, indexing, and schematic presentations. (Of course, I may not be smart enough to know what to ask!)

ChiraTech Scientific Instruments has created a board that should, in my opinion, be in every well-equipped computer. I heartily recommend it to your attention. This review is woefully too short (of necessity) to explain even a few of the capabilities of this fine creation. I will be happy to discuss the board with anyone who calls after 7:00 PM Central Time.

CSC,Inc.XASM..MACRO Sets.Review

By Steven M. Ward
39 Thorndike St.
Arlington, MA 02174

I will review some useful and cost-effective software, the cross-assembler macro sets by Computer Systems Consultants, Inc. These macro sets provide cross-assembler capability for a FLEX9 6809 microcomputer host environment for 6800/6801/6802/6808, 6803, 6805, 6502, 8080/8085, and Z-80 target microcomputers. Before presenting the review of this software, I will briefly present a background to cross-machine software development in order to provide a perspective for the personal computer user.

Cross-machine software development

Often a programmer finds himself in a working environment which requires writing software for more than one type, model, or brand of computer or microprocessor. One solution is to have a separate machine of each type for software development for that particular computer or microprocessor. Another solution is to provide the necessary software tools for software development for all of the varying computers and microprocessors in one uniform environment for the programmer, which means offering these software tools on a single machine.

There are considerations of economy, efficiency, etc. In choosing between these two environments. Certainly for the personal user the latter approach is the most desirable. This kind of software development environment is usually referred

ARTISAN SYSTEMS CORP.

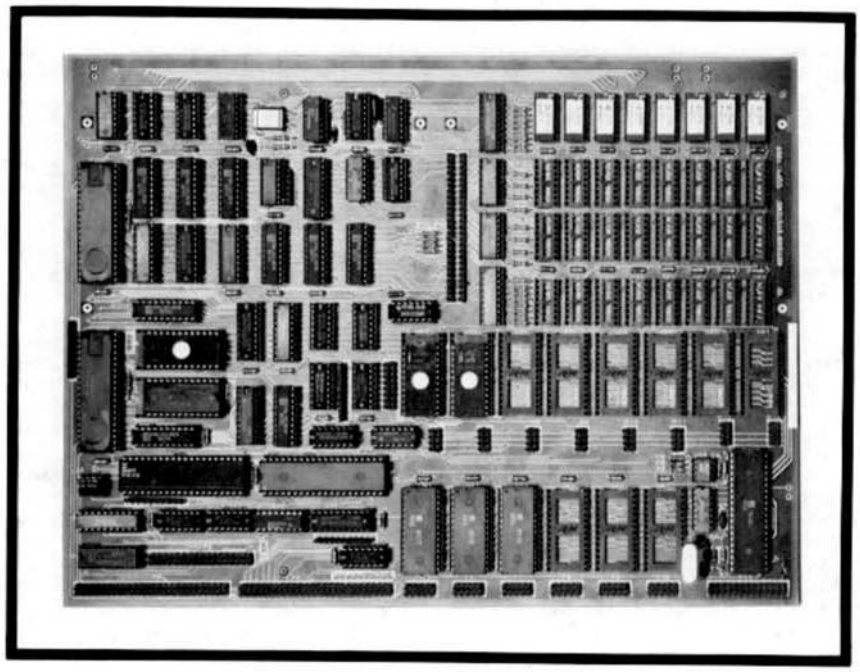
DP-09
6809 BASED SINGLE BOARD SYSTEM

2 MHZ OPERATION

256K DRAM

6 RS-232 SERIAL PORTS

8 28-PIN SOCKETS



DUAL 68B09E PROCESSORS

FLOPPY CONTROLLER

WINCHESTER INTERFACE

4 LAYER PC BOARD

FEATURES:

- 68B09E advanced 8/16 bit system processor with memory management hardware allows for one megabyte address space.
- 64K - 256K byte DRAM.
- 8 each 28-pin sockets for up to 128K byte EPROM. EPROMS can be 2732, 2764 or 27128. In addition 2K x 8 or 8K x 8 static RAMS may be used.
- Six RS-232 serial ports with full modem handshake. Advanced 6551A ACIAs with software baud rate select of 110 to 19.2K baud.
- 6522A interface chip provides two 16-bit timers plus two 8-bit parallel ports (unbuffered)
- 8 auto-vectored interrupts for high speed I/O handling
- 50-pin expansion socket
- Independent 68B09E subsystem for disk control 1-4 floppys 5" or 8" DS, SD, DD. SASI interface allows 5-45 MBYTE Winchesters to be connected using external controller. The subsystem uses a proprietary DMA technique for high speed operation.
- FLEX operating system is available.

PRICES

DP-09 A&T 64K, 4 Serial Ports	**$895**
FLEX for DP-09	$150
OS-9 Level One & Two	CALL
Forth	CALL

ARTISAN SYSTEMS CORP.
410 CROSS STREET
WINCHESTER, MA 01890
(617) 721-2109

ALL ORDERS PREPAID. VISA & MASTERCARD ACCEPTED. 3-4 WEEKS DELIVERY. (ADD 2 WEEKS FOR PERSONAL CHECKS)
FLEX™ is a trademark of Technical Systems Consultants. OS-9™ is a trademark of Microware.

to as "cross-machine" support. For example, an assembler for a 6502 microprocessor which actually runs on a 6809 computer is called a "cross-assembler". More than a cross-assembler is needed to provide a software development capability for a "target" machine on a different "host" machine.

In general, a simulator or emulator is provided which allows the object code of the target computer to be executed on the host machine via the simulator. Usually the simulator provides execution trace and debugging facilities to the programmer. The disadvantage is that all testing is not done in a real-time environment which means that the simulator cannot duplicate the actual speed and timing characteristics of the target computer though it can simulate all of the logical functions of the target machine.

Besides the simulator, cross-development aids such as disassemblers, eprom programmers, etc. may be found on a host system computer which provide software development capability for various target machines. It is clear that only a cross-assembler and an editor are needed on a host machine to allow the programmer to write an assembly language program for a target machine, assemble that program, and generate executable object code for the target machine. But it is also clear that a simulator for the target machine is highly desirable since very few programs work properly on the first try.

In this "cross-development" environment the programmer can use the host system's editor and utilities to write his code for a target machine. The host machine is of course the machine he is actually using and the target machine is the computer for which the program is actually intended.

The programmer will then assemble his program using the cross-assembler for the target machine. Naturally, his program is assembly language mnemonics for the target machine and the output of the cross-assembler will be object code which is executable on the target machine. The programmer is faced with the task of physically transporting his object code from his host machine to his target machine, and this may be done via eprom or by some other means. First, the programmer will need to test and debug his program. Since the programmer has a simulator for his target machine he will simply execute his target machine object code via the simulator for the purposes of testing and debugging and then he will transport his code when he is satisfied his program works.

6800/1, 6805, and 6502 CSC cross-assemblers

Computer Systems Consultants, inc. sells cross-assemblers for other microprocessors as previously mentioned, but this review is confined to the microprocessors for which I have CSC cross-assemblers. Since all of the microprocessor cross-assemblers are constructed similarly, this discussion applies to the other microprocessor cross-assemblers except that I have only tested these particular cross-assemblers.

Now that I have used the name "cross-assembler" freely, let me say that CSC does not sell cross-assemblers per-se. A cross-assembler would truly be a stand-alone assembler running on a host machine which accepts mnemonics and generates object code for a target machine. CSC sells sets of macros for use with the Technical Systems Consultants 6809 macro assembler which together provide cross-assembler functionality. A different macro set is provided for each different target microprocessor. To effectively utilize these cross-assembler macro sets you must have the TSC 6809 macro assembler, though since the macro set is necessarily source code, it is possible to adapt the macro set to another macro assembler. Version 2 of the TSC 6809 macro assembler is needed for use with these macros.

In addition to the sets of macros there is one program called the macro translator which is needed to perform cross-assembly. Briefly, for a FLEX9 environment using the CSC cross-assembler macros you need:

1) TSC 6809 macro assembler, version 2.

2) CSC macro translator program.

3) CSC macro set for the target microprocessor.

Of course, you obviously also need to write an assembly language source code program for the target machine so that you will have a FLEX9 source code file waiting to be converted into object code for the target microprocessor by the above software.

The macro translator is in effect a preprocessor which converts the source code file into a form which is compatible with the TSC macro assembler. For instance, a source code program written for the 8080 microprocessor has a different format from a 6809 source code program, yet the programmer would like to write his 8080 source code in a manner entirely compatible with an 8080 microprocessor environment. With the CSC macro translator the programmer can write his source code in the target machine format. This source file is then processed by the macro translator which generates another source file for input to the TSC 6809 macro assembler. Here are the steps:

1) Prepare a target machine source code assembly language file.

2) Input the target source code file to the macro translator which outputs an intermediate source code file suitable for the TSC 6809 macro

3) Input the macro translator intermediate source code file to the TSC 6809 macro assembler which outputs a binary object file for execution on the target machine.

The purpose of the macro translator is to allow the programmer to write a target machine assembly language source code program that is entirely compatible with "native" assemblers for the target machine and yet ultimately utilize the TSC 6809 macro assembler to perform the cross-assembly. The macro translator comes from CSC in source code form and has to be assembled by the user. It is written to be assembled on either a FLEX2 6800 machine or a FLEX9 6809 machine and will adjust itself accordingly to the two environments. It is not necessary for the user to modify the macro translator for either environment. The macro translator functions with all of the macro sets so that there is only one version of the macro translator.

CSC cross-assembler documentation

The printed documentation provided with the cross-assembly software is excellent. The documentation clearly explains the "how-to" aspects of user operation and also explains the functions performed by the macro translator for each of the various microprocessors. The documentation notes all target machine assembly language source code file requirements such as not allowing line numbers in 8080/8085 source code files, and a few other restrictions, none of which conflict with common assembly language environments for the target microprocessors.

Example files are provided on the software disk. These files include a target machine assembly language source code file and the subsequent macro translator source code file for each target microprocessor for which you purchase cross-assembler macro sets. These example files are useful, but disappointing. They are essentially nonsense source code files and you have to place the two files side-by-side and study them to begin to see what the macro translator is doing in processing the original source code into the intermediate source code. It would have been much better to provide a very simple source code program like an ACIA INCH or OUTCH routine, or several of these exceedingly simple routines to demonstrate the macro translator processing. Certainly all of the macro translator functions are demonstrated via the nonsense text files provided, but I think most people prefer to study "real" source code.

CSC cross-assembler operation

To actually use the cross-assembler macros and the macro translator is very simple. Just execute the macro translator (MACXLAT.CMD) and it will prompt for input and output file names. The input file needs a LIB statement at the beginning of the source code for the corresponding macro set. If you are processing code for a 6502 microprocessor then you need to add "LIB MAC6502" to the front of the original source code file. The intermediate source code file output by the macro translator may be immediately assembled using the TSC 6809 macro assembler. The TSC assembler will of course generate a binary object file which is executable binary code for the target microprocessor. That is all there is to it!

I received the macro sets for the 6800/1, 6805, and 6502 micros. I assembled several old 6800 programs and everything went smoothly. Emboldened by this success I turned the software over to a couple of associates at least as butter-fingered as yours truly and asked them to generate some 6805 and 6502 programs. Several programs straight out of some textbooks along with a couple of original programs were produced in short order. The only problems encountered were the normal typos and logic failures associated with these endeavors, but most importantly, the cross-assemblers produced the correct responses in all cases. Certainly this testing was not exhaustive, but the use of these macro sets continues with satisfied users at this end.

Conclusion

Computer Systems Consultants, Inc. of Conyers, Georgia offers an excellent product in these cross-assembler macros. I hope to review additional software from CSC in the near future. In particular, I have received the CSC Super Sleuth suite of programs which perform sophisticated disassembly and editing of binary files and a few other nice functions. I'm sure some of you would like a review of simulators which are generally referred to in this article. Let me hear your comments and suggestions.

RECORD LEVEL LOCKOUT..OS9

ADDING A STORM DOOR

Jeff Seng
Mark Krall
Intercomp Company
2320 Xenium Lane
Plymouth ,Mn. 55441
612-559-0446

The OS-9 (level 2) operating system is quite possibly the most sophisticated OS for microcomputers available today. One frustration with OS-9 has been the long wait for the record level lockouts which were promised and advertised over 2 years ago. The lockouts were recently made available in a "final release" in early 1983.

We have found that the OS-9 lockouts are not totally dependable under heavy usage, at least within BASIC09. We have developed a technique to dramatically increase the efficacy of the lockouts thru the use of a "STORM DOOR" file, thus allowing their use in actual business multi-user database applications.

In a typical business application, two or more terminal operators are entering transaction reports into a transaction file (See Fig.1). Other operators are reading from this file to prepare accounts payable and receivable reports etc. If there were no lockouts, then different transactions from the inputting operators might be overwritten. With the OS-9 lockouts, this indeed happens infrequently, perhaps once every 300 updates with our application package. This is unacceptable for serious use.

It is tempting to make use of the OS-9 file attribute choices by making the transaction file non-sharable to prevent this overwriting. Since the actual disc updates take very little time, the application program could include an error trap to recognize that the non-shareable file was busy, wait a little while and try again. The operator would notice very little delay. Unfortunatley this would prevent input when other operators were scanning thru the transaction file even on a read only basis, to prepare, say, an accounts receivable report.

It would be ideal if there was a file attribute that would make the file sharable for reads, but would only allow one user to make updates while the file was open. Of course, this fantasy attribute doesn't exist in OS-9. The trick is to create a "STORM DOOR" file that is non-sharable. This storm door file is ignored by all processes that read the transaction file but all processes that might desire to write to the transaction file are first required to go thru the "storm door". This metaphor comes from the Minnesota tundra practice of adding a second (storm) door to the front door in times of cold weather.

The programs that update the transaction file are slightly modified so that the storm door file must be opened for an update before the transaction file is updated (see Fig.2). If the storm door file is busy then the program sleeps a few ticks and tries again. After writing to the transaction file the program closes the storm door thus allowing other users to input data.

The probability of an overwrite should now be the product of the probabilities of the record level lockout failures and the file level lockout failures. Thus the probability of an overwrite should be about 1/300 of the probability of a non-sharable file lockout failure which should be small indeed. In fact, we have noticed no overwrites to date in systems using this storm door algorithm.

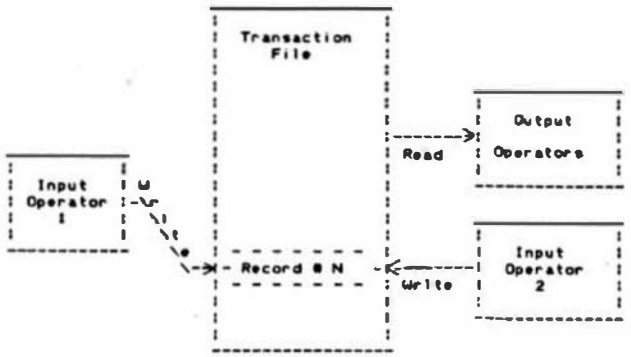


Figure 1

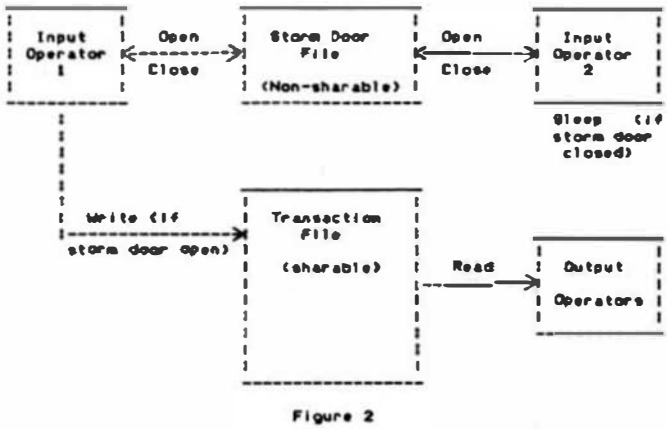


Figure 2

THE COMPLETE BUSINESS SYSTEM

+Multiuser+Highly Expandable+Cost Effective

S+ THE CONCEPT

The S+ system is a modular computer system in which all portions of the hardware and software are designed to work together in the most efficient way possible. An S+ single user system with floppy disk storage is a competitive and cost effective entry level system. Unlike most other small computers being sold as "personal", or "small business" machines, the S+ system may be expanded to maximum capabilities using this same hardware and software. You cannot end up with a DEAD END system that cannot be expanded and whose software is not compatible with larger machines. A basic S+ system may be expanded to thirty-two users, a megabyte of main memory and hundreds of megabytes of hard disk storage by simply plugging in, or connecting the desired upgrade equipment.

TOTAL DESIGN—Hardware and Software

The S+ system is an integrated hardware and software design. The two complement and enhance each other in this system. The UniFLEX® operating system used in the S+ systems is patterned after the Bell Laboratories UNIX® operating system, one of the most admired and widely used operating systems in the world. Instead of being an afterthought, the software is part of the design of the S+ system. You can be sure that with this approach that all parts of the computer operate with maximum efficiency and cost effectiveness.

THE CENTRAL PROCESSOR

The basic S+ system is configured with 256K bytes of memory and can be expanded to more than 1 million bytes. An efficient and fast hardware memory management system is used to allocate the available memory among the users on a dynamic basis. As little as 8K bytes, or the entire memory—if needed—can be used by any individual user. This makes it possible to run very large programs on the system, but it also uses no more memory than necessary for a particular job. The increase in cost effectiveness of this system over crude and outdated bank switching arrangements is dramatic.

The central processor runs in both user and supervisor states. It can detect and reject a defective user program. It is impossible for a user program to go bad and stop the entire system, as can happen quite easily in less sophisticated systems.

Task switching is accomplished by use of a multiple map RAM memory, with sixty-four individual task maps. Each task can access from 4 to 64 K-bytes of memory. Multiple tasks may be used in programs that require more than 64K bytes of memory for execution. When a task is completed the memory is automatically released for other use.

SOFTWARE

The S+ operating system, UniFLEX® is a multiuser, multitasking operating system based on the UNIX® operating system that has been used for many years on Digital Equipment Corp. PDP-11 series minicomputers. It is considered one of the most sophisticated and "user friendly" operating systems available. Variations of UNIX® are rapidly becoming standard on mini and larger microcomputers.

A large variety of languages are available for use with the system. These include FORTRAN, COBOL, BASIC, and Pascal. Word processing packages are also available to give you full text processing capability on the system.

Applications programs are available in large quantities in many fields. This includes general business, medical, dental, veterinary, library and real estate management; plus others. Since the system is multiuser it can also be connected to cash registers to produce a point-ofsale terminal system combined with the computer. The possibilities for application of this system are endless.

THE I/O SYSTEM

The S+ system is totally interrupt driven. All terminal and printer I/O devices connect to an I/O bus separate from the main bus. Up to thirty-two separate devices may be connected to the I/O bus at any one time. If I/O activity is great enough to cause an unacceptable slowdown in system operation, a separate I/O processor can be installed in the system. This plug-in option removes all I/O handling overhead from the main processor and allows operation of up to thirty-two external devices at 9,600 baud. Without an integrated total design, as in the S+ system, it would become impractical to use a UNIX®type operating system in a situation with heavy terminal I/O activity.

DISK STORAGE

A wide range of disk storage capacity is available for the S+ system, from 2.5 M-byte floppy disks to an 80 M-byte Winchester and many sizes between. All disk controllers use direct memory access (DMA) type operations to maximize data transfer and to minimize overhead on the main processor. The Winchester disks also use intelligent controllers along with DMA transfers to preserve the performance that these type devices are capable of giving. Without this distributed intelligence the system performance would be greatly degraded. The UniFLEX® operating system is designed to work at maximum efficiency with this type disk system. The data transfer rates achieved by this combination rival those of large minicomputers.

COMMUNICATIONS

A high speed local network communications system is available to interconnect S+ systems. The VIABUS® network will allow communication between systems at data rates of over 400K baud. Such a system makes it possible to share data between local systems in an efficient and low-cost manner.

AVAILABLE SOON

Tape backup—20M-Byte in less than 15 minutes on a standard ¼ inch cartridge.

Mini-Wini—5 and 10 M-Byte Winchesters—5¼ inch package. Winchester performance, for smaller systems in a small package. UniFLEX® compatible design.

Large Capacity—190 and 340 M-Byte Winchesters, plus SMD cartridge drives.

UniFLEX is a registered trademark of Technical Systems Consultants, Inc.

UNIX is a registered trademark of Bell Labs.

VIABUS is a registered trademark of Southwest Technical Products Corporation.

PISTON SIM.-CT82-BASIC&ASM

U N I V E R S I T Y O F T R A N S K E I

Chemistry Department
Private Bag X5092
U TATA
Republic of Transkei

June 1st., 1983

Computer Publishing Center
68 Micro Journal
5900 Cassandra Smith
P.O. Box 849
Hixson, TN 37343
USA

Piston Simula or for CT-82 terminal.

Dear Sir,

I must congratulate your recent contributor Ron Raine (68' Micro, March 1983, p40) for his excellent graphics display program. The program is a pleasure to watch and I often run it when the computer is idle. Ron complains that no CT-82 graphics programs have appeared in your journal, so I would like to contribute something to this effect.

The program PISTON.BAS runs on a CT-8212 terminal under TSC XBASIC, and simulates the piston action of one cylinder of a typical 4-stroke engine. I have inserted enough comments to make the program intelligible without too much clutter. Of course all the REM statements can be dispensed with, and probably some improvements could be made as a program is never really finished!

The lines 1600-1700 set up the graphic modes used by the program. I have created a file containing the most commonly used graphic commands for the CT-82, as a subroutine, which I can then append to any Basic program and delete the lines not used. This saves a lot of repetetive typing.

The other program PISTON.TXT,(the object file is PISTON.CMD) is the source code for the analogous program in assembly language. I initially wrote this version to see if there would be any speed improvement, hence instead of calculating the screen data, as most of the Basic program does, I decided to let the processor read the data as a long stream and print it with little or no computation. I reasoned that this would be the faster method. However, except for the 'spark' and 'explosion', which are very much faster in machine code, the display speed is nearly the same or slightly slower than the Basic program!

I want to mention a trick which might be useful to other readers in general. I needed to have the long strings of numeric data for the assembly language program, but I was not prepared to type in all those numbers in the FCB statements, especially since any typing error would have made a mess of the display and would have been tricky to track down. Therefore I decided to use the Basic program PISTON.BAS generate the data on disk files. An example will illustrate the procedure.

To generate the data for the 'FILHD' block edit PISTON.BAS as follows:

```
104 T$=",":REM comma, data separator
105 OPEN NEW "FILHD" AS 1
106 PRINT #1,"FILHD " : GOTO 640

640 FOR J%=1 TO 30
645 PRINT #1," FCB ";
650 PRINT #1,SL$+T$+CHR$(X%+J%)+T$+CHR$(Y%)+T$+CHR$(X%+J%)+T$+CHR$(V%)
660 NEXT J%
665 PRINT #1," FCB 255" : CLOSE 1 : STOP
```

By doing this for all the graphic modules, one has as set of files with the proper format, labels and pseudo-ops for the assembly language file. It is only necessary to insert each file into the proper place using an editor (I prefer Stylograph for this) or have them inserted at assembly time using the LIB pseudo-op.

Yours faithfully,

Dr. Lorenzo Placenza, Senior Research Associate.

```
10 REM ************ PISTON.BAS ************
20 REM written 1983. by L.P.L. Placenza,
30 REM Chemistry Dept., University of Transkei,
40 REM Umtata, Transkei, Southern Africa.
50 REM ****************************************
60 REM For a smooth EXIT, type ^C and any key.
70 ON ERROR GOTO 1500
80 GOSUB 1610
90 PRINT $6
100 PRINT CHR$(5):REM NO CURSOR
110 REM ***** FRAME *****
120 FOR J=1 TO 4:READ A,B,C,D
130 PRINT SL$+CHR$(A)+CHR$(B)+CHR$(C)+CHR$(D);
140 NEXT J
150 DATA 0,0,184,0,184,0,184,66,184,66,0,66,0,66,0,0
160 PRINT CU$+CHR$(28)+CHR$(2);
170 PRINT "SIMULATED 4-STROKE PISTON SEQUENCE"
180 PRINT CU$+CHR$(28)+CHR$(31)
190 PRINT "----^----^----^----^----^----^----^----"
200 PRINT $S$
210 REM ***** DRAW WALLS *****
220 A$=20:AY=20:AU=130:AV=20
230 GOSUB 870
240 AY=50:AV=50
250 GOSUB 870
260 REM ***** SPARKPLUG *****
270 R=31:REM NUMBER OF DATA PAIRS TO READ
280 FOR I=1 TO R
290 READ H,V
300 PRINT $D$+CHR$(H)+CHR$(V);
310 NEXT I
320 DATA 130,33,131,33,131,34,131,35
330 DATA 131,36,131,37,132,34,133,34
340 DATA 134,34,135,34,136,34,137,34
350 DATA 132,36,133,36,134,36,135,36
360 DATA 136,36,137,36,137,35,129,33
370 DATA 128,33,127,33,129,32,128,37
380 DATA 127,37,127,36,127,35,127,34
390 DATA 126,36,125,36,125,35
400 A=130:AY=20:AV=130:AV=50
410 GOSUB 870
420 REM ***** EXHAUST VALVE DATA *****
430 E3=130:F3=24:E4=140:F4=24
440 GOSUB 900
450 PRINT CU$+CHR$(73)+CHR$(8);
460 PRINT "EXHAUST VALVE"
470 REM ***** INLET VALVE DATA *****
480 E3=130:F3=44:E4=140:F4=44
490 GOSUB 900
500 GOSUB 990:GOSUB 1080
510 PRINT CU$+CHR$(73)+CHR$(15)
520 PRINT "INLET VALVE"
530 REM ***** BD SHAFT *****
540 A=1:AY=38:AU=22:AV=38
550 GOSUB 870
560 A=1:AY=40:AU=22:AV=40
570 GOSUB 870
580 REM ***** BD PISTON HEAD *****
590 T=20:A=-1:B=19
600 I=22:F=21:V=49:I=30
610 CY=1:REM STARTING CYCLE NUMBER
620 GOSUB 1160
630 REM ***** FILL HEAD *****
640 FOR J=1 TO 30
650 PRINT SL$+CHR$(I+J)+CHR$(T)+CHR$(I+J)+CHR$(V);
660 NEXT J
670 REM ***************** MAIN LOOP *****************
680 REM ***** UPSTROKE *****
690 FOR LL=0 TO 65
700 PRINT SL$+CHR$(I+LL)+CHR$(Y)+CHR$(I+LL)+CHR$(V)
710 PRINT SL$+CHR$(I+LL+T0)+CHR$(Y)+CHR$(I+LL+T0)+CHR$(V);
720 PRINT CL$+CHR$(I+LL+A)+CHR$(Y)+CHR$(I+LL+A)+CHR$(V);
730 PRINT SD$+CHR$(I+LL-1)+CHR$(Y+9);:REM SHAFT EXTENSION
740 PRINT SD$+CHR$(I+LL-1)+CHR$(V-9);
750 NEXT LL
760 GOSUB 1150
770 REM ***** DOWNSTROKE *****
780 FOR LL=65 TO 0 STEP -1
790 PRINT SL$+CHR$(I+LL+A):+CHR$(Y)+CHR$(I+LL+A)+CHR$(V)
800 PRINT SL$+CHR$(I+LL+A)+CHR$(Y)+CHR$(I+LL+A)+CHR$(V);
810 PRINT CL$+CHR$(I+LL+T0)+CHR$(Y)+CHR$(I+LL+T0)+CHR$(V)
820 NEXT LL
830 GOSUB 1150
840 GOTO 690
850 REM ***************** END OF MAIN LOOP *****************
860 D$=CHR$(0):PRINT LF$:D$;D$;D$;D$;:END:REM PROVIDE FOR SMOOTH EXIT.
870 PRINT SL$+CHR$(A)+CHR$(AY)+CHR$(AU)+CHR$(AV);
880 RETURN
890 REM ***** DRAW VALVE *****
900 PRINT SL$+CHR$(E3)+CHR$(F3)+CHR$(E4)+CHR$(F4);
910 RETURN
920 REM ***** CLOSE EXHAUST VALVE *****
930 F1=22:F2=24
940 FOR EI=125 TO 120
950 PRINT CL$+CHR$(EI)+CHR$(F1)+CHR$(EI)+CHR$(F2);
960 PRINT SL$+CHR$(EI+1)+CHR$(F1)+CHR$(EI+1)+CHR$(F2)
970 NEXT EI:RETURN
980 REM ***** OPEN INLET VALVE *****
990 F3=44:F4=48:F1=44
1000 FOR EI=130 TO 126 STEP -1
1010 PRINT CL$+CHR$(EI)+CHR$(F3)+CHR$(EI)+CHR$(F4);
1020 PRINT SD$+CHR$(EI)+CHR$(F1)
1030 PRINT SL$+CHR$(EI-1)+CHR$(F3)+CHR$(EI-1)+CHR$(F4);
1040 NEXT EI:RETURN
1050 REM ***** OPEN EXHAUST VALVE *****
1060 F3=26:F4=22:F1=26
1070 GOTO 1000
1080 REM ***** CLOSE INLET VALVE *****
1090 F3=44:F4=48
1100 FOR EI=125 TO 120
1110 PRINT CL$+CHR$(EI)+CHR$(F3)+CHR$(EI)+CHR$(F4);
1120 PRINT SL$+CHR$(EI+1)+CHR$(F3)+CHR$(EI+1)+CHR$(F4);
1130 NEXT EI:RETURN
1140 REM ***** CYCLE COUNT *****
1150 PRINT CU$+CHR$(4)+CHR$(20);"CYCLE";CY;
1160 IF CY=3 THEN PRINT $00;"E X H A U S T  S T R O K E      "
1170 IF CY=1 THEN PRINT $00;"C O M P R E S S I O N  S T R O K E"
1180 IF CY=4 OR CY=0 THEN PRINT $00;"F U E L / A I R  I N          "
1190 IF CY=2 THEN PRINT $00;"P O W E R  S T R O K E          "
1200 IF CY=2 THEN CY=0:GOSUB 1200:GOSUB 1620
```

```
1210 CY1=CY1+1
1220 IF CY1=1 THEN GOSUB 930:GOSUB 990
1230 IF CY1=4 THEN GOSUB 1040
1240 IF CY1=2 THEN GOSUB 1090
1250 IF CY1=5 THEN CY1=1:GOTO 1220
1260 RETURN
1270 REM ***** SPARK *****
1280 FOR K=1 TO 10
1290 RESTORE 1390
1300 FOR J=1 TO 8
1310 READ HS,VS
1320 PRINT SD$+CHR$(HS)+CHR$(VS);
1330 NEXT J
1340 RESTORE 1390
1350 FOR J=1 TO 8
1360 READ HS,VS
1370 PRINT CD$+CHR$(HS)+CHR$(VS);
1380 NEXT J:NEXT K:RETURN
1390 DATA 124,35,123,34,122,34,122,33
1400 DATA 123,33,124,32,125,31,126,32
1410 REM ***** EXPLOSION *****
1420 FOR X=1 TO 5
1430 RESTORE 1530
1440 FOR J=1 TO 29
1450 READ BH,BV
1460 PRINT SD$+CHR$(BH)+CHR$(BV);
1470 NEXT J
1480 RESTORE 1530
1490 FOR J=1 TO 29
1500 READ BH,BV
1510 PRINT CD$+CHR$(BH)+CHR$(BV);
1520 NEXT J:NEXT K:RETURN
1530 DATA 124,23,126,35,125,26,126,29,128,29
1540 DATA 125,33,127,44,126,48,126,26,122,29
1550 DATA 123,34,124,22,125,39,121,39,121,25
1560 DATA 126,46,125,44,126,34,127,43,128,23
1570 DATA 116,44,116,23,117,33,117,38,118,29
1580 IF ERR=26 THEN RESUME 860:REM CONTROL C TRAP.
1590 ON ERROR GOTO:END
1800 REM ***** GRAPHICS CONSTANTS *****
1610 CU$=CHR$(11):'REM SET CURSOR X,Y
1620 ES$=CHR$(30)+CHR$(24):REM DISABLE SCROLL ON LF
1630 LF$=CHR$(30)+CHR$(8):REM RE-ENABLE SCROLL ON LF
1640 GR$=CHR$(29)+CHR$(22):REM GRAPHICS MODE SET
1650 SL$=CHR$(29)+CHR$(3):'REM SET LINE
1660 SD$=CHR$(29)+CHR$(19):'REM SET DOT
1670 CD$=CHR$(29)+CHR$(20):'REM CLEAR DOT
1680 CL$=CHR$(29)+CHR$(4):'REM CLEAR LINE
1690 CO$=CU$+CHR$(27)+CHR$(20)
1700 RETURN

* H-STROKE PISTON SIMULATION
* For a SWTc 6212 terminal only.
*
* PISTON.TXT
*
* Exit using ESCAPE. Will exit when piston next
* reaches left side after ESC key pressed.
*
* FLEX6 equates - GIMIX FLEX & GRIMAS-09 V2.0.

SYSFCB  EQU   $CB40
WARMS   EQU   $CD03
GETCHR  EQU   $CD15
PUTCHR  EQU   $CD18
PSTRNG  EQU   $CD1E
PCRLF   EQU   $CD24
OUTDEC  EQU   $CD39
STAT    EQU   $CD4E
FMS     EQU   $D406
INCHNE  EQU   $F804 inPut, no echo to terminal.

* GRAPHIC COMMAND EQUATES

SL EQU $1003 set line
CL EQU $1004 clear line
SD EQU $1013 set dot
CD EQU $1014 clear dot
CU EQU 11 set cursor position X,Y

 ORG 0

PISTON BRA START

VER FCC 001,'.','.001 version number
 FCC ', (c) 1983 by L.P.L. Piacenza'

START LEAX TITLE,PCR
 JSR PSTRNG
 JSR INCHNE

* GO TO GRAPHICS MODE

 LEAX GRAPH,PCR
 LBSR DRAW
 LEAX CURSOR,PCR disable cursor
 LBSR DRAW

* FRAME

 LEAX FRAME,PCR
 LBSR DRAW

* HEADING

 LEAX HEADNG,PCR
 LBSR DRAW

* WALLS

 LEAX WALLS,PCR
 LBSR DRAW

* PLUG

 LEAX PLUG,PCR
 LDY #31
SPLOOP
 LDB #58
 LBSR ISUB
 LDA 0,X+
 JSR PUTCHR
 LDA 0,X+
 JSR PUTCHR
 LEAY -1,Y
 BNE SPLOOP

* EXHAUST VALVE

 LEAX XS90E,PCR
 LBSR DRAW
 LEAX OPEX,PCR
 LBSR DRAW

* INLET VALVE

 LEAX XS90I,PCR
 LBSR DRAW
 LEAX CLOSIN,PCR
 LBSR DRAW

* SHAFT

 LEAX SHAFT,PCR
 LBSR DRAW

* FILL PISTON HEAD

 LEAX FILHD,PCR
 LBSR DRAW

* Main PART - PISTON MOVEMENT

 BSR TESTCY
MAIN
 LBSR KSTAT
 CMPA #'9
 LBEQ EXIT
 CMPA #27
 LBEQ EXIT

* UPSTROKE

 LEAX UP8,PCR
 LBSR DRAW
 BSR TESTCY

* DOWN STROKE
 LEAX DOWN8,PCR
 LBSR DRAW
 BSR TESTCY
 BRA MAIN

* END OF MAIN LOOP.

* TEST CYCLE NUMBER 'CY'

TESTCY
 BSR ISTRSS
 LDA CY,PCR
 CMPA #2
 BNE NEF
 LDY #10
SPRES LEAX SPARKS,PCR
 LBSR DRAW
 DECB
 BNE SPRES
 LDB #5
BANG LEAX EXPLOS,PCR
 LBSR DRAW
 DECB
 BNE BANG
NEF
 LDA CY,PCR
 INCA
 STA CY,PCR
TESTC2
 LDA CY,PCR
 CMPA #1
 BNE NEF1
 LBSR SCLEI
 LBSR SOPIN
NEF1 LDA CY,PCR
 CMPA #4
 BNE NEF2
 BSR SOPEI
NEF2
 LDA CY,PCR
 CMPA #2
 BNE NEF22
 BSR SCLIN
NEF22
 LDA CY,PCR
 CMPA #5
 BNE NEF3
 LDA #1
 STA CY,PCR
 BRA TESTC2
NEF3
 STA CY,PCR
 RTS

ISTRSS
 LEAX CYCLE,PCR
 BSR DRAW
 LEAX CC+,PCR
 CLRB
 JSR OUTDEC
 LDB CY,PCB
 CMPB #3
 BNE TST2
 LEAX NEI,PCR
 BRA DRAW

TST2
 CMPB #1
 BNE TST3
 LEAX NCOMP,PCR
 BRA DRAW

TST3
 CMPB #4
 BNE TST4
 LEAX NFUEL,PCR
 BRA DRAW

TST4
 CMPB #2
 BNE TST5
 LEAX NPOWER,PCR
 BRA DRAW

TST5 RTS

EXIT
 LEAX ENABLE,PCR
 BSR DRAW
 JMP WARMS

* OUTPUT A STRING TERMINATED BY $FF.

DRAW: JSR PUTCHR
DRAW LDA 0,X+
 CMPA #$FF
 BNE DRAW1
 RTS

ISUB JSR PUTCHR
 EXG A,B
 JMP PUTCHR

SCLEI LEAX CLEI,PCR close exhaust
 BRA DRAW

SOPEI LEAX OPEI,PCR open exhaust
 BRA DRAW

SOPIN LEAX OPENIN,PCR open inlet
 BRA DRAW

SCLIN LEAX CLOSIN,PCR close inlet
 BRA DRAW

* KEYBOARD STATUS - test for escape key.
 KSTAT
  LDY #500
 KLOOP JSR STAT
  BEQ NOKEY
  JMP INCHNE

NOKEY LEAY -1,Y
 BNE KLOOP
 RTS

* DATA

GRAPH FCB 29,22,255
CURSOR FCB 30,21,30,24,255

ENABLE FCB 30,0,21,255

FRAME
 FCB SL
 FCB 0,0,184,0
 FCB SL
 FCB 184,0,184,66
 FCB SL
 FCB 184,66,0,66
 FCB SL
 FCB 0,66,0,0,255

WALLS FCB SL
 FCB 20,20,130,20
 FCB SL
 FCB 20,50,130,50
 FCB SL
 FCB 130,20,130,50
 FCB SL
 FCB 130,24,140,24
 FCB SL
 FCB 130,44,140,44,255

PLUG
 FCB 130,33,131,33,131,34,131,35,131,36
 FCB 131,37,132,34,133,34,134,34,135,34
 FCB 130,34,137,34,132,36,133,36,134,36
 FCB 135,36,136,36,137,36,137,35,129,33
 FCB 128,33,127,33,129,37,128,37,127,37
 FCB 127,36,127,35,127,34,124,36,125,36
 FCB 125,35

SHAFT
 FCB SL
 FCB 1,30,22,30
 FCB SL
 FCB 1,40,22,40,255

EXH
 FCB SL
 FCB 130,24,140,24
 FCC 11,73,0,"EXHAUST VALVE",255

INL
 FCB SL
 FCB 130,46,140,44
 FCC 11,73,15,"INLET VALVE",255

CLEI
 FCB 29,4,125,22,125,26,29,3,126,22,126,26
 FCB 29,0,126,22,126,26,26,3,127,22,127,26
 FCB 29,4,127,22,127,26,29,3,128,22,128,26
 FCB 29,4,128,22,128,26,29,3,129,22,129,26
 FCB 255

OPENIN
 FCB 29,0,130,44,130,48,29,19,130,44
 FCB 29,3,129,44,129,48,29,4,129,44,129,48
 FCB 29,19,129,46,29,3,128,44,128,48,29,4
 FCB 128,44,128,48,29,19,128,46,29,3,127,44
 FCB 127,48,29,0,127,44,127,48,29,19,127,46
 FCB 29,3,126,44,126,48,29,4,126,46,126,48
 FCB 29,19,126,46,29,3,125,44,125,48
 FCB 255

OPEI
 FCB 29,4,130,26,130,22,29,19,130,24
 FCB 29,3,129,26,129,22,29,4,129,26,129,22
 FCB 29,19,129,24,29,3,128,26,128,22,29,4
 FCB 128,26,128,22,29,19,128,24,29,3,127,26
 FCB 127,22,29,4,127,24,127,22,29,19,127,24
 FCB 29,3,126,26,126,22,29,0,126,26,126,22
 FCB 29,19,126,24,29,3,125,26,125,22
 FCB 255

CLOSIN
 FCB 29,4,125,44,125,48,29,3,126,44,126,48
 FCB 29,4,126,44,126,48,29,3,127,44,127,48
 FCB 29,4,127,44,127,48,29,3,128,44,128,48
 FCB 29,4,128,44,128,48,29,3,129,44,129,48
 FCB 255

XS90E FCC 11,73,0,"EXHAUST VALVE",255
XS90I FCC 11,73,15,"INLET VALVE",255

TITLE FCB 12
 FCC "PRESS ANY KEY TO START",4
```

```
HEADING FCC 11,28,2,'SIMULATED 4-STROKE PISTON'
 FCC ' SEQUENCE'
      FCC 11,28,3,'----------'
 FCC '----------',255

MSG1 FCC 11,27,20,'E X H A U S T '
 FCC 'S T R O K E                ',255
MSG2 FCC 11,27,20,'C O M P R E S S I O N'
 FCC '  S T R O K E              ',255
MFUEL FCC 11,27,20,'F U E L / A I R  I N'
 FCC '                           ',255
MPOWER FCC 11,27,20,'P O W E R  S T R O K E'
 FCC '                           ',255
CYCLE FCC 11,4,20,'CYCLE ',255

PIL40
 FCB 29,3,23,24,23,40,29,3,24,21,24,40
 FCB 29,3,25,21,25,40,29,3,26,21,26,40
 FCB 29,3,27,21,27,40,29,3,28,23,28,40
 FCB 29,3,29,21,29,40,29,3,30,21,30,40
 FCB 29,3,31,21,31,40,29,3,32,21,32,40
 FCB 29,3,33,21,33,40,29,3,34,21,34,40
 FCB 29,3,35,21,35,40,29,3,36,23,36,40
 FCB 29,3,37,21,37,40,29,3,38,21,38,40
 FCB 29,3,39,21,39,40,29,3,40,21,40,40
 FCB 29,3,41,21,41,40,29,3,42,21,42,40
 FCB 29,3,43,21,43,40,29,3,44,21,44,40
 FCB 29,3,45,21,45,40,29,3,46,21,46,40
 FCB 29,3,47,21,47,40,29,3,48,21,48,40
 FCB 29,3,49,21,49,40,29,3,50,21,50,40
 FCB 29,3,51,21,51,40,29,3,52,21,52,40
 FCB 255

UPS
 FCB 29,3,22,21,22,40,29,3,50,21,50,40
 FCB 29,4,21,21,21,40,29,19,21,30,29,19
 FCB 21,40
 FCB 29,3,25,21,25,40,29,3,51,21,51,40
 FCB 29,4,22,21,22,40,29,19,22,30,29,19
 FCB 22,40
 FCB 29,3,24,21,24,40,29,3,52,21,52,40
 FCB 29,4,23,21,23,40,29,19,23,30,29,19
 FCB 23,40
 FCB 29,3,25,21,25,40,29,3,53,21,53,40
 FCB 29,4,24,21,24,40,29,19,24,30,29,19
 FCB 24,40
 FCB 29,3,26,21,26,40,29,3,54,21,54,40
 FCB 29,4,25,21,25,40,29,19,25,30,29,19
 FCB 25,40
 FCB 29,3,27,21,27,40,29,3,55,21,55,40
 FCB 29,4,26,21,26,40,29,19,26,30,29,19
 FCB 26,40
 FCB 29,3,28,21,28,40,29,3,56,21,56,40
 FCB 29,4,27,21,27,40,29,19,27,30,29,19
 FCB 27,40
 FCB 29,3,29,21,29,40,29,3,57,21,57,40
 FCB 29,4,28,21,28,40,29,19,28,30,29,19
 FCB 28,40
 FCB 29,3,30,21,30,40,29,3,58,21,58,40
 FCB 29,4,29,21,29,40,29,19,29,30,29,19
 FCB 29,40
 FCB 29,3,31,21,31,40,29,3,59,21,59,40
 FCB 29,4,30,21,30,40,29,19,30,30,29,19
 FCB 30,40
 FCB 29,3,32,21,32,40,29,3,60,23,60,40
 FCB 29,4,31,21,31,40,29,19,31,30,29,19
 FCB 31,40
 FCB 29,3,33,21,33,40,29,3,61,21,61,40
 FCB 29,4,32,21,32,40,29,19,32,30,29,19
 FCB 32,40
 FCB 29,3,34,21,34,40,29,3,62,21,62,40
 FCB 29,4,33,21,33,40,29,19,33,30,29,19
 FCB 33,40
 FCB 29,3,35,21,35,40,29,3,63,23,63,40
 FCB 29,4,34,21,34,40,29,19,34,30,29,19
 FCB 34,40

 FCB 29,3,36,21,36,40,29,3,64,21,64,40
 FCB 29,4,35,21,35,40,29,19,35,30,29,19
 FCB 35,40
 FCB 29,3,37,21,37,40,29,3,65,21,65,40
 FCB 29,4,36,21,36,40,29,19,36,30,29,19
 FCB 36,40
 FCB 29,3,38,21,38,40,29,3,66,21,66,40
 FCB 29,4,37,21,37,40,29,19,37,30,29,19
 FCB 37,40
 FCB 29,3,39,21,39,40,29,3,67,21,67,40
 FCB 29,4,38,21,38,40,29,19,38,30,29,19
 FCB 38,40
 FCB 29,3,40,21,40,40,29,3,68,21,68,40
 FCB 29,4,39,21,39,40,29,19,39,30,29,19
 FCB 39,40
 FCB 29,3,41,23,41,40,29,3,69,21,69,40
 FCB 29,4,40,21,40,40,29,19,40,30,29,19
 FCB 40,40
 FCB 29,3,42,21,42,40,29,3,70,23,70,40
 FCB 29,4,41,21,41,40,29,19,41,30,29,19
 FCB 41,40
 FCB 29,3,43,21,43,40,29,3,71,21,71,40
 FCB 29,4,42,21,42,40,29,19,42,30,29,19
 FCB 42,40
 FCB 29,3,44,21,44,40,29,3,72,21,72,40
 FCB 29,4,43,23,43,40,29,19,43,30,29,19
 FCB 43,40
 FCB 29,3,45,21,45,40,29,3,73,21,73,40
 FCB 29,4,44,21,44,40,29,19,44,30,29,19
 FCB 44,40
 FCB 29,3,46,21,46,40,29,3,74,21,74,40
 FCB 29,4,45,21,45,40,29,19,45,30,29,19
 FCB 45,40
 FCB 29,3,47,21,47,40,29,3,75,21,75,40
 FCB 29,4,46,21,46,40,29,19,46,30,29,19
 FCB 46,40
 FCB 29,3,48,21,48,40,29,3,76,23,76,40
 FCB 29,4,47,21,47,40,29,19,47,30,29,19
 FCB 47,40
 FCB 29,3,49,21,49,40,29,3,77,21,77,40
 FCB 29,4,48,21,48,40,29,19,48,30,29,19
 FCB 48,40
 FCB 29,3,50,21,50,40,29,3,78,21,78,40
 FCB 29,4,49,21,49,40,29,19,49,30,29,19
 FCB 49,40
 FCB 29,3,51,21,51,40,29,3,79,21,79,40
 FCB 29,4,50,21,50,40,29,19,50,30,29,19
 FCB 50,40
 FCB 29,3,52,21,52,40,29,3,80,21,80,40
 FCB 29,4,51,21,51,40,29,19,51,30,29,19
 FCB 51,40
 FCB 29,3,53,21,53,40,29,3,81,21,81,40
 FCB 29,4,52,21,52,40,29,19,52,30,29,19
 FCB 52,40
 FCB 29,3,54,24,54,40,29,3,82,21,82,40
 FCB 29,4,53,21,53,40,29,19,53,30,29,19
 FCB 53,40
 FCB 29,3,55,21,55,40,29,3,83,21,83,40
 FCB 29,4,54,21,54,40,29,19,54,30,29,19
 FCB 54,40
 FCB 29,3,56,21,56,40,29,3,84,21,84,40
 FCB 29,4,55,21,55,40,29,19,55,30,29,19
 FCB 55,40
 FCB 29,3,57,21,57,40,29,3,85,21,85,40
 FCB 29,4,56,21,56,40,29,19,56,30,29,19
 FCB 56,40
 FCB 29,3,58,21,58,40,29,3,86,21,86,40
 FCB 29,4,57,21,57,40,29,19,57,30,29,19
 FCB 57,40
 FCB 29,3,59,21,59,40,29,3,87,21,87,40
 FCB 29,4,58,24,58,40,29,19,58,30,29,19
 FCB 58,40
 FCB 29,3,60,21,60,40,29,3,88,23,88,40
 FCB 29,4,59,21,59,40,29,19,59,30,29,19
 FCB 59,40
 FCB 29,3,61,21,61,40,29,3,89,23,89,40
 FCB 29,4,60,21,60,40,29,19,60,30,29,19
 FCB 60,40
 FCB 29,3,62,21,62,40,29,3,90,21,90,40
 FCB 29,4,61,21,61,40,29,19,61,30,29,19
 FCB 61,40
 FCB 29,3,63,21,63,40,29,3,91,21,91,40
 FCB 29,4,62,21,62,40,29,19,62,30,29,19
 FCB 62,40
 FCB 29,3,64,21,64,40,29,3,92,21,92,40
 FCB 29,4,63,21,63,40,29,19,63,30,29,19
 FCB 63,40
 FCB 29,3,65,21,65,40,29,3,93,21,93,40
 FCB 29,4,64,21,64,40,29,19,64,30,29,19
 FCB 64,40
 FCB 29,3,66,21,66,40,29,3,94,21,94,40
 FCB 29,4,65,21,65,40,29,19,65,30,29,19
 FCB 65,40
 FCB 29,3,67,21,67,40,29,3,95,21,95,40
 FCB 29,4,66,21,66,40,29,19,66,30,29,19
 FCB 66,40
 FCB 29,3,68,21,68,40,29,3,96,21,96,40
 FCB 29,4,67,21,67,40,29,19,67,30,29,19
 FCB 67,40
 FCB 29,3,69,23,69,40,29,3,97,21,97,40
 FCB 29,4,68,21,68,40,29,19,68,30,29,19
 FCB 68,40
 FCB 29,3,70,21,70,40,29,3,98,21,98,40
 FCB 29,4,69,21,69,40,29,19,69,30,29,19
 FCB 69,40
 FCB 29,3,71,21,71,40,29,3,99,21,99,40
 FCB 29,4,70,21,70,40,29,19,70,30,29,19
 FCB 70,40
 FCB 29,3,72,21,72,40,29,3,100,21,100,40
 FCB 29,4,71,21,71,40,29,19,71,30,29,19
 FCB 71,40
 FCB 29,3,73,21,73,40,29,3,101,21,101,40
 FCB 29,4,72,21,72,40,29,19,72,30,29,19
 FCB 72,40
 FCB 29,3,74,21,74,40,29,3,102,21,102,40
 FCB 29,4,73,21,73,40,29,19,73,30,29,19
 FCB 73,40
 FCB 29,3,75,21,75,40,29,3,103,21,103,40
 FCB 29,4,74,21,74,40,29,19,74,30,29,19
 FCB 74,40
 FCB 29,3,76,21,76,40,29,3,104,21,104,40
 FCB 29,4,75,21,75,40,29,19,75,30,29,19
 FCB 75,40
 FCB 29,3,77,21,77,40,29,3,105,21,105,40
 FCB 29,4,76,21,76,40,29,19,76,30,29,19
 FCB 76,40
 FCB 29,3,78,21,78,40,29,3,106,21,106,40
 FCB 29,4,77,21,77,40,29,19,77,30,29,19
 FCB 77,40
 FCB 29,3,79,21,79,40,29,3,107,21,107,40
 FCB 29,4,78,21,78,40,29,19,78,30,29,19
 FCB 78,40
 FCB 29,3,80,21,80,40,29,3,108,21,108,40
 FCB 29,4,79,21,79,40,29,19,79,30,29,19
 FCB 79,40
 FCB 29,3,81,21,81,40,29,3,109,21,109,40
 FCB 29,4,80,21,80,40,29,19,80,30,29,19
 FCB 80,40
 FCB 29,3,82,21,82,40,29,3,110,21,110,40
 FCB 29,4,81,21,81,40,29,19,81,30,29,19
 FCB 81,40
 FCB 29,3,83,21,83,40,29,3,111,21,111,40
 FCB 29,4,82,21,82,40,29,19,82,30,29,19
 FCB 82,40
 FCB 29,3,84,21,84,40,29,3,112,21,112,40
 FCB 29,4,83,21,83,40,29,19,83,30,29,19
 FCB 83,40
 FCB 29,3,85,21,85,40,29,3,113,21,113,40
 FCB 29,4,84,21,84,40,29,19,84,30,29,19
 FCB 84,40
 FCB 29,3,86,21,86,40,29,3,114,21,114,40
 FCB 29,4,85,21,85,40,29,19,85,30,29,19
 FCB 85,40
 FCB 29,3,87,21,87,40,29,3,115,21,115,40
 FCB 29,4,86,21,86,40,29,19,86,30,29,19
 FCB 86,40,255

DOWNS
 FCB 29,3,86,21,86,40,29,3,114,21,114,40
 FCB 29,4,115,21,115,40,29,3,85,21,85,40
 FCB 29,3,113,21,113,40,29,4,114,21,114,40
 FCB 29,3,84,21,84,40,29,3,112,23,112,40
 FCB 29,4,113,21,113,40,29,3,83,21,83,40
 FCB 29,3,111,21,111,40,29,4,112,21,112,40
 FCB 29,3,82,21,82,40,29,3,110,21,110,40
 FCB 29,4,111,21,111,40,29,3,81,21,81,40
 FCB 29,3,109,21,109,40,29,4,110,21,110,40
 FCB 29,3,80,21,80,40,29,3,108,23,108,40
 FCB 29,4,109,21,109,40,29,3,79,21,79,40
 FCB 29,3,107,21,107,40,29,4,108,21,108,40
 FCB 29,3,78,21,78,40,29,3,106,21,106,40
 FCB 29,4,107,21,107,40,29,3,77,21,77,40
 FCB 29,3,105,21,105,40,29,4,106,21,106,40
 FCB 29,3,76,21,76,40,29,3,104,21,104,40
 FCB 29,4,105,21,105,40,29,3,75,21,75,40
 FCB 29,3,103,21,103,40,29,4,104,21,104,40
 FCB 29,3,74,21,74,40,29,3,102,21,102,40
 FCB 29,4,103,21,103,40,29,3,73,21,73,40
 FCB 29,3,101,21,101,40,29,4,102,21,102,40
 FCB 29,3,72,21,72,40,29,3,100,23,100,40

 FCB 29,4,101,21,101,40,29,3,71,21,71,40
 FCB 29,3,99,21,99,40,29,4,100,21,100,40
 FCB 29,3,70,21,70,40,29,3,98,21,98,40
 FCB 29,4,99,21,99,40,29,3,69,21,69,40
 FCB 29,3,97,21,97,40,29,4,98,21,98,40
 FCB 29,3,68,21,68,40,29,3,96,21,96,40
 FCB 29,4,97,21,97,40,29,3,67,21,67,40
 FCB 29,3,95,21,95,40,29,4,96,21,96,40
 FCB 29,3,66,21,66,40,29,3,94,21,94,40
 FCB 29,4,93,21,95,40,29,3,65,21,65,40
 FCB 29,3,93,21,93,40,29,4,94,23,94,40
 FCB 29,3,64,21,64,40,29,3,92,21,92,40
 FCB 29,4,93,21,93,40,29,3,63,21,63,40
 FCB 29,3,91,21,91,40,29,4,92,21,92,40
 FCB 29,3,62,21,62,40,29,3,90,23,90,40
 FCB 29,4,91,21,91,40,29,3,61,21,61,40
 FCB 29,3,89,21,89,40,29,4,90,23,90,40
 FCB 29,3,60,21,60,40,29,3,88,21,88,40
 FCB 29,4,89,21,89,40,29,3,59,21,59,40
 FCB 29,3,87,21,87,40,29,4,88,21,88,40
 FCB 29,3,58,21,58,40,29,3,86,23,86,40
 FCB 29,4,87,21,87,40,29,3,57,21,57,40
 FCB 29,3,85,21,85,40,29,4,86,21,86,40
 FCB 29,3,56,21,56,40,29,3,84,28,84,40
 FCB 29,4,85,21,85,40,29,3,55,21,55,40
 FCB 29,3,83,21,83,40,29,4,84,21,84,40
 FCB 29,3,54,21,54,40,29,3,82,21,82,40
 FCB 29,4,83,23,83,40,29,3,53,21,53,40
 FCB 29,3,81,21,81,40,29,4,82,21,82,40
 FCB 29,3,52,21,52,40,29,3,80,21,80,40
 FCB 29,4,81,21,81,40,29,3,51,21,51,40
 FCB 29,3,79,21,79,40,29,4,80,21,80,40
 FCB 29,3,50,21,50,40,29,3,78,21,78,40
 FCB 29,4,79,21,79,40,29,3,49,21,49,40
 FCB 29,3,77,21,77,40,29,4,78,21,78,40
 FCB 29,3,48,21,48,40,29,3,76,21,76,40
 FCB 29,4,77,21,77,40,29,3,47,21,47,40
 FCB 29,3,75,21,75,40,29,4,76,21,76,40
 FCB 29,3,46,23,46,40,29,3,74,21,74,40
 FCB 29,4,75,21,75,40,29,3,45,21,45,40
 FCB 29,3,73,21,73,40,29,4,74,21,74,40
 FCB 29,3,44,21,44,40,29,3,72,21,72,40
 FCB 29,4,73,21,73,40,29,3,43,21,43,40
 FCB 29,3,71,21,71,40,29,4,72,21,72,40
 FCB 29,3,42,23,42,40,29,3,70,21,70,40
 FCB 29,4,71,21,71,40,29,3,41,21,41,40
 FCB 29,3,69,21,69,40,29,4,70,21,70,40
 FCB 29,3,40,21,40,40,29,3,68,21,68,40
 FCB 29,4,69,25,69,40,29,3,39,21,39,40
 FCB 29,3,67,21,67,40,29,4,68,21,68,40
 FCB 29,3,38,21,38,40,29,3,66,21,66,40
 FCB 29,4,67,21,67,40,29,3,37,21,37,40
 FCB 29,3,65,21,65,40,29,4,66,21,66,40
 FCB 29,3,36,21,36,40,29,3,64,21,64,40
 FCB 29,4,65,21,65,40,29,3,35,23,35,40
 FCB 29,3,63,21,63,40,29,4,64,21,64,40
 FCB 29,3,34,21,34,40,29,3,62,21,62,40
 FCB 29,4,63,21,63,40,29,3,33,21,33,40
 FCB 29,3,61,21,63,40,29,4,62,21,62,40
 FCB 29,3,32,21,32,40,29,3,60,21,60,40
 FCB 29,4,61,21,61,40,29,3,31,21,31,40
 FCB 29,3,59,21,59,40,29,4,60,21,60,40
 FCB 29,3,30,21,30,40,29,3,58,21,58,40
 FCB 29,4,59,21,59,40,29,3,29,21,29,40
 FCB 29,3,57,21,57,40,29,4,58,21,58,40
 FCB 29,3,28,21,28,40,29,3,56,21,56,40
 FCB 29,4,57,21,57,40,29,3,27,21,27,40
 FCB 29,3,55,21,55,40,29,4,56,21,56,40
 FCB 29,3,26,21,26,40,29,3,54,21,54,40
 FCB 29,4,55,21,55,40,29,3,25,21,25,40
 FCB 29,3,53,21,53,40,29,4,54,21,54,40
 FCB 29,3,24,21,24,40,29,3,52,21,52,40
 FCB 29,4,53,21,53,40,29,3,23,21,23,40
 FCB 29,3,51,21,51,40,29,4,52,21,52,40
 FCB 29,3,22,23,22,40,29,3,50,21,50,40
 FCB 29,4,51,21,51,40,29,3,21,21,21,40
 FCB 29,3,49,21,49,40,29,4,50,21,50,40
 FCB 255

SPARKS
 FCB 29,19,124,33,29,19,123,34
 FCB 29,19,122,34,29,19,122,33
 FCB 29,19,123,33,29,19,124,32
 FCB 29,19,125,33,29,19,120,32
 FCB 29,20,124,33,29,20,125,34
 FCB 29,20,122,34,29,20,122,33
 FCB 29,20,123,33,29,20,124,32
 FCB 29,20,125,33,29,20,126,32
 FCB 255

EXPLOS
 FCB 29,19,120,23,29,19,126,33
 FCB 29,19,125,26,29,19,126,27
 FCB 29,19,120,29,29,19,125,33
 FCB 29,19,127,44,29,19,126,48
 FCB 29,19,120,26,29,19,122,25
 FCB 29,19,123,34,29,19,120,22
 FCB 29,19,125,30,29,19,121,39
 FCB 29,19,121,33,29,19,126,46
 FCB 29,19,125,44,29,19,126,34
 FCB 29,19,127,43,29,19,120,23
 FCB 29,19,116,44,29,19,116,23
 FCB 29,19,117,33,29,19,117,38
 FCB 29,19,118,29,29,20,124,23
 FCB 29,20,126,33,29,20,125,26
 FCB 29,20,126,27,29,20,120,29
 FCB 29,20,125,33,29,20,127,44
 FCB 29,20,126,48,29,20,120,26
 FCB 29,20,122,25,29,20,123,34
 FCB 29,20,124,22,29,20,125,39
 FCB 29,20,121,39,29,20,121,33
 FCB 29,20,126,44,29,20,123,44
 FCB 29,20,126,34,29,20,127,43
 FCB 29,20,120,23,29,20,116,44
 FCB 29,20,116,23,29,20,117,33
 FCB 29,20,117,38,29,20,118,29
 FCB 255

* CONSTANTS

CCT FCB 9
CY FCB 3

 END PISTON
```

WHEW!

BIT BUCKET

BLOOPER – OURS

In the March 1983 issue of 68 MICRO JOURNAL, page 26 we (meaning 'I') left off some lines of a **Modem Program**.

Here is the missing code – please accept my apology!!

```
451  )B2DD 1026 0039          LBNE   FILERR
452   B2E1 20   EF            BRA    DISLOP
453
454   B2E3 B6   04    FILCLO   LDA    #4
455   B2E5 A7   84             STA    ,X
456   B2E7 BD   D406           JSR    FMS        CLOSE FILE
457   B2EA 7E   CD03           JMP    WARMS
```


Computer Systems Consultants, Inc.
1454 Lotta Lane, Conyers, GA 30207

E. A. (Bud) Pass, Ph. D.

Dear Dan: 05/13/83

I would like to comment on several letters to the editor which appeared in the June 1983 issue of '68' Micro Journal, and one which appeared in an earlier issue.

Jim Schrader comments on a "bug" in the TSC Extended Basic interpreter. He apparently does not understand the concept of roundoff. In his example, he assumes that the expression

```
    1/3*3
```

should be equal to

```
    1*3/3
```

which, given infinite precision, it would be. However, consider the computation to be done in 17 digit arithmetic with X equal to three, and left to right evaluation of equal priority operators. Then

```
    1/3*3 = .33333333333333333*3 = .99999999999999999
```

and

```
    1*3/3 = 3/3 = 1
```

providing unequal results without using any assumption except for limited precision.

The extra computations in the "fix" simply obscure the roundoff issue and cause additional roundoff to again produce the value 1 from the modified expression. The output conversion algorithm prints "1" rather than "0.99999999999999999" because it is programmed to round off, not truncate, values it is outputting. The Unifloc version of TSC Basic addresses the problem of roundoff with the "approximately equal" relational operator "~=", which produces a result similar to the following expression:

```
    if abs((1-r)<eps then ...
```

where "eps" is a small value denoting the acceptable level of roundoff.

Thus Jim's assertion of the discovery of a "bug" in TSC Basic is not justified.

TSC Basic does have a true, serious bug, in the "IF" statement. If more than one statement appears between the "then" and the "else", any statements after the "else" will never be performed, even if the conditional expression is false. Thus, in

```
    If i=2 then print "i=2":print "i=2" else print "i<>2"
```

nothing will be printed. This error has been validated in several of the latest versions of TSC Basic, and was reported several years ago to TSC, without success.

Adrianus Hoogenraad inquired about the Motorola 6800 User's Group. It has not been operational since the late 1970's; I don't know the exact date or reasons for its discontinuance, but it provided a useful function while it lasted. He also commented on the excellence and low price of ASI boards. I agree on the excellence of the boards; however, the low price was due entirely to the fact that ASI is withdrawing from the SS-50 marketplace, and the boards were previously not that inexpensive at all.

Robert Van Houten suggests a rather cumbersome approach to dumping hard disks to floppy diskettes. I encounter the same problem often when copying files from 8" floppies to 5.25" floppies. The standard FLEX COPY program makes the task quite manageable. However, the replacement for the COPY program provided by Leo Taylor provides several features which make the effort much easier. It has options to alphabetize the list of files being copied, prompt before copying each file, copy only files which do not exist on the target disk, copy only files which do exist on the target disk (but are older), etc. In addition, it refuses to copy a file which will not fit onto the target disk. Since Leo provides the COPY program (and others, such as an enhanced CAT utility, etc.) free, I suggest that Mr. Van Houten contact Leo Taylor before getting out his coding machine again and coding up a list of file names.

In an earlier issue of '68' Micro Journal, someone was questioning how to get a modem to work with a SWTPC MP-S2 serial board. The problem with the MP-S2 board is that SWTPC uses a nonstandard scheme of wiring the interface connector, which is compatible with the CT-82, but is not directly compatible with anything else.

The MP-S2 provides the following lines on each of its DB-25 connectors:

Line name	Pin	I/O
Chassis Ground	1	COM
Received Data	2	OUT
Transmitted Data	3	IN
---	4/5	I/O
Signal Ground	7	COM
Data Carrier Det	8	OUT
Sec Data Carr Det	12	IN
Hold Down	19	IN
Clear to Send	20	IN
Clock In	24	IN

RS-232 uses an voltage protocol in which a positive voltage between +3 and +25 volts represents a logical false state and a negative voltage between -3 and -25 volts represents a logical true state. Voltages falling into the nonsense range of -3 to +3 may be interpreted as being in either state. One signal ground is used as a reference for all data and control lines.

Following is a list of the standard RS-232-C circuits, from the point of view of a terminal (as opposed to a computer or modem):

Circuit identification	Line	Source
Chassis Ground	1	COM
Transmitted Data	2	DTE
Received Data	3	DCE
Request To Send	4	DTE
Clear To Send	5	DCE
Data Set Ready	6	DCE
Signal Ground	7	COM
Data Carrier Detect	8	DCE
Positive Test Voltage	9	DCE
Negative Test Voltage	10	DCE
Equalizer Mode	11	DCE
Sec Data Carrier Detect	12	DCE
Sec Clear To Send	13	DCE
Sec Transmitted Data	14	DTE
Transmitter Clock	15	DCE
Sec Received Data	16	DCE
Receiver Clock	17	DCE
Receiver Digit Clock	18	DCE
Sec Request To Send	19	DTE
Data Terminal Ready	20	DTE
Signal Quality Detect	21	DCE
Ring Indicator	22	DCE
Data Rate Selector	23	DTE
Ext Transmitter Clock	24	DTE
Busy	25	DTE

To modify this table to represent the point of view of a modem or computer, and not a terminal, the sense of all receivers and transmitters would be reversed; in particular, lines 2-3, 4-5, 8-20, 15-17, etc.

To connect an MP-S2 to a terminal other than a CT-82, the only modification required (if any) may be to built pin 20 to +12 thru a 1k ohm resistor. Some terminals have their own, non-standard, conflicting usage which may require that the extra lines on the MP-S2 not be connected to the corresponding lines on the terminal. However, lines 2, 3, and 7 should always be connected respectively to the same lines on the terminal. Also, it may be necessary to cut the leads to pins 8, 12, 19, 20, and 24 and reconnect the old lead to pin 8 to pin 5, the old lead to pin 12 to pin 8, the old lead to pin 20 to pin 4, and move the pullup resistor from pin 20 to pin 4. This will standardize the connection on the MP-S2 side.

Hooking a modem to an MP-S2 generally requires reversing the lines mentioned above (2-3, 4-5, 8-20, etc.). If the change is permanent, it may be easier to change the port on the MP-S2 and use a straight-thru cable; otherwise, a custom cable must be constructed to reverse the lines.

Sincerely,

E. A. (Bud) Pass

DMS NOTES..UPDATE

Bill Adams
21 Larch Road
Briarcliff, N.Y. 10510

Several readers have inquired as to what happened to the mailing list article I referred to in the February '83 issue. Actually, I wasn't too happy with the way it turned out and asked Larry Williams to hold off print until I could revise it. I had forgotton that it was referenced by the checkbook article. Needless to say, here it is, revised and updated, with apologies to those who searched back issues for it.

One of the most common applications for a computer system is that of a mailing list. It is also an appropriate starting point for the new user since it is a relatively simple application, consisting of a single file. In examples which follow, I will use the new XDMS Data Management System to further simplify matters. The DMS2/VM system could also be used with some variations.

The first compulsion in designing a mailing list file is to define fields for first and last names, address, city, state and zip code. We may also wish to assign each person a number for later reference. Such a file definition might appear as follows:

```
|ADDRESS            FILE--- PRINT--
|# GRP/FLD  FMT    OFS LEN OFS LEN
|1 ADDRESS  C,7
|  ACCT     N,4      0   2   0   4
|  LAST     A,12     2  12   5  12
|  FIRST    A,10    14  10  18  10
|  ADDRESS  A,18    24  18  29  18
|  CITY     A,18    42  18  48  18
|  ST       A,2     60   2  67   2
|  ZIP      A,5     62   5  70   5
|                   ----------
|TOTAL RECORD LENGTH:  67      75
```

Sample records within such a file might appear as follows when input or printed. Note: The names and addresses are fictitious. Any resemblance to real people is coincidental.

```
|ADDRESS
|ACCT LAST------ FIRST---- ADDRESS------- CITY---------- ST ZIP--
|0001 Anderson   Harold B. 34 Prescott Dr. Brooklyn      NY 10030
|0002 Benson     Roger D.  1 Water Street  Peekskill     NY 10570
| :
| etc.
```

There are both advantages and drawbacks to a fixed field format as just described. One advantage is the ability to select records on any of the fields defined. This permits, for example, printing of all names and addresses within a certain city or state. Another advantage is the ability to sort by any of the file fields.

One drawback is the fixed format itself. If data which exceeds the field width is to be entered, it must be abbreviated. Further, we did not provide for additional address information, such as company apartment number, title ("Mr.","Mrs.","Dr.") etc. Certainly, we could define fields for each of these data, but would likely end up with a record which exceeds the screen or printer width. With XDMS this is not a grave problem, since the record may be "folded" into two or more lines. File space may be wasted, however, by unfilled fields.

Physical mailing labels may be generated with XDMS by utilizing the GENERATE facility. A control file resembling the following will produce labels for each file record on 1" labels at six lines/inch.

```
|MARGIN 0,0 PAGE 6,0, NO-HEADING, PAUSE OFF
|FILE ADDRESS LIST, TYPESET TEXT
|FORM
|<FIRST><LAST>
|<ADDRESS>
|<CITY><ST><ZIP>
|ENDX
|END
```

The PAGE command sets the output page size to six lines, The LIST command causes all fields to be selected for output, the TEXT command suppresses the normal tabular output and the TYPESET command causes suppression of multiple spaces within the FORM lines which follow. The FORM shown consists of the desired mailing label fields and the ENDX command terminates the FORM image.

There is an alternative format. If we define a file with non-specific address fields we can then accomodate almost any type of address data. We do however sacrifice the capability to select and/or sort on specific fields. Let's examine such a file format using our previous data.

```
|ADDRESS LIST
|ACCT # ADDRESS----------
|0001 1 Harold B. Anderson
|     2 34 Prescott Drive
|     3 Brooklyn, N.Y.    10030
|0002 1 Roger D. Benson
|     2 1 Water Street
|     3 Peekskill, N.Y.   10570
|     :
|   etc.
```

This format also offers both advantages and drawbacks. The overwhelming advantage, of course, is that a variable number of lines may be used to accomodate the address data, and being non-specific, any fields (name, company, etc.) may be entered as required. This "vertical" format incidentally, is employed in the Westchester ABS General Accounting System where up to four lines depict both balance sheet accounts as well as customers and suppliers.

A "class" code is used to classify various accounts.

The drawback of this format is that reference to individual fields is lost. The user, thus, must make a determination as to which format best suits his/her needs. A compromise design would be to define (in our minds) number assignments (# field) for first and last names, company, tel#, etc. This permits omission of line data without wasting file space, and also permits recreation of a file which resembles our original example. This may be done with "...ARRAY ADDRESS FOR # = 1,2,3,4..." and "...SUM BY ACCT..." instructions input into the GENERATE facility. The resulting output consists of an array record for each ACCT value with fixed sized fields labeled "1","2"... containing the ADDRESS data. This file may be subsequently input to another process to print labels, forms, etc.

Label generation using the "vertical" format where actual label images are stored (eg; file example above) may be accomplished with the following:

```
|MARGIN 0,0, PAGE 6,0, PAUSE OFF
|FILE ADDRESS, NO-HEADING, NO-DATA-LABEL,
|IF # => 1, IF # <= 6,
|CACHE ACCT, PRINT ADDRESS
|ON ACCT EJECT
|END
```

Here, records with a "#" value from 1 to 6 are selected and the ADDRESS field printed. The CACHE

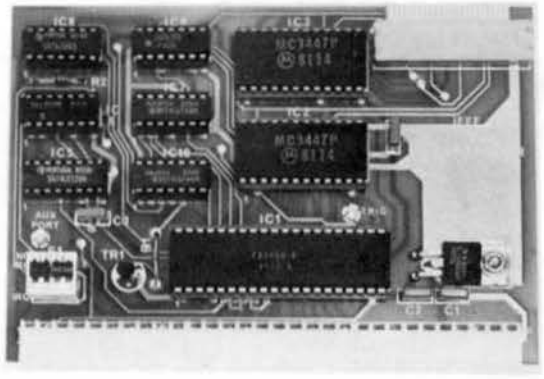
WINDRUSH MICRO SYSTEMS

UNIVERSAL EPROM PROGRAMMER

- PROGRAMS and VERIFIES 2508, 2708, 2516, 2716, 2532, 2732A, 2564, and 2764 EPROMS. Minor hardware mods are required to program the INTEL 27128.
- Tri-volt and Single Volt 2508/2708 and 2516/2716 devices are supported.
- ZIF sockets with mode selector switches eliminate 'personality modules'.
- Twin boards with five feet of twisted pair planar cable puts the programmer out on the bench where it belongs.
- SS-30 and EXORCISOR interfaces are available.
- Menu driven software provides the following facilities:
  a. MOVE blocks of memory within the buffer.
  b. READ an EPROM into the buffer.
  c. VERIFY an EPROM against the buffer.
  d. EXAMINE and change the contents of the buffer.
  e. DUMP the contents of the buffer in HEX and ASCII.
  f. FILL a selected area of the buffer with a specified character.
- Software available for all versions of SSB DOS, FLEX 2, FLEX 9 and OS-9. Assembly language source files supplied on disk....enables customizing.
- Well documented users manual provides step-by-step adaptation and operating instructions.

AVAILABLE FROM GIMIX IN THE U.S.A.

IEEE-488



- SUPPORTS ALL PRINCIPAL MODES OF THE IEEE-488 (1975/8) BUS SPECIFICATION:
  - Talker
  - Listener
  - System Controller
  - Serial Poll
  - Parallel Poll
  - Group Trigger
  - Single or Dual Primary Address
  - Secondary Address
  - Talk only...Listen only
- Fully documented with a complete reprint of the KILOBAUD article on the IEEE bus.
- Low level assembly language drivers suitable for 6800, 6801, 6802, 6803, 6808 and 6809 are supplied in the form of listings. These drivers have been extensively tested and are GUARANTEED to work!
- Single SS-30 board (4, 8, or 16 addresses per port), fully socketed, gold plated bus connectors, and IEEE interface cable assembly.

PL/9

EDITOR/COMPILER/DE-BUGGER

- Friendly inter-active environment where you have INSTANT access to the Editor, the Compiler, and the Trace-Debugger, which, amongst other things, can single step the the program a SOURCE line at a time. You also have direct access to any FLEX utility and your System Monitor.
- 250 page manual is organized as a tutorial with plenty of examples.
- Fast single pass compiler produces 8K of COMPACT and FAST 6809 machine code output per minute with no run-time overheads or license fees.
- Fully compatible with TSC text editor format disk files
- Signed and unsigned BYTEs and INTEGERs, 32-bit floating point REALs.
- Vectors (single dimension arrays) and Pointers are supported.
- Mathematical expressions: (+), (-), (*), (/), modulus (%), negation (-)
- Expression evaluators: (=), (<>), (<), (>), (>=), (<=)
- Bit operators: (AND), (OR), (EOR/XOR), (NOT), (SHIFT), (SWAP)
- Logical operators: (.AND), (.OR), (.EOR/.XOR).
- Control statements: IF..THEN..ELSE, IF..CASE1..CASE2..ELSE, BEGIN..END, WHILE.., REPEAT..UNTIL, REPEAT..FOREVER, CALL, JUMP, RETURN, BREAK, GOTO.
- Direct access to (ACCA), (ACCB), (ACCD), (CCR) and (XREG).
- FULLY supports the MC6809 SWI, SWI2, SWI3, NMI, FIRQ, IRQ and RESET vectors. Writing a self-starting (from power-up) program that uses ANY, or ALL, of the MC6809 interrupts is an absolute snap!
- Procedures may be passed and may return variables. This makes them functions which behave as though they were an integral part of PL/9.
- Several fully documented library function modules are supplied: IOSUBS, BETIO, HARDIO, HEXIO, FLEXIO, SCIPACK, STRSUBS, and REALCOM.

'... THIS IS THE MOST EFFECIENT COMPILER I HAVE FOUND TO DATE.'

Quoted from Ron Anderson's FLEX User Notes column. Need we say more?

MACE/XMACE

A co-resident EDITOR/ASSEMBLER for the 6809 written by Graham Trott which takes most of the pain out of assembly language program development:

- Friendly inter-active environment where you have INSTANT access to the Editor, the Assembler, FLEX and your System Monitor.
- MACE can also produce ASMPROC's for PL/9 with the assembly language source passed to the output file as comments.
- Includes XMACE a co-resident 6800/1/2/3/8 EDITOR/CROSS ASSEMBLER.

'C'

This is the FLEX version of the James McCosh 'C' compiler that is also available on UNIFLEX from SWTP and OS-9 from Microware:

- The FLEX implementation supports the full Kernighan and Ritchie 'C' specification except 'floats', 'doubles', and 'bit-fields'.
- Produces very efficient assembly language source output with the 'C' source optionally interleaved as comments.
- Built-in optimizer will shorten object code by about 11%
- Supports interleaved assembly language programs.
- The TSC relocating assembler/linking loader (SPO9-17) is REQUIRED.

MACE	(6809 FLEX only).....	$ 98.00
PL/9	(6809 FLEX only)...(a steal at this price!).....	$198.00
'C'	(a 56K system and the TSC SPO9-17 package is req'd)....	$295.00
IEEE-488	with IEEE-488 cable assembly.....	$298.00
SS-30	Universal EPROM programmer w/one version of software...	$375.00
EXORciser	Universal EPROM programmer w/one version of software...	$395.00
SOFTWARE	drivers for a 2nd, 3rd, or 4th operating system.....	$ 25.00

ALL PRICES INCLUDE AIR MAIL POSTAGE

WE STOCK THE FOLLOWING COMPANIES PRODUCTS: GIMIX, SSB, FHL, MICROWARE, TSC, LUCIDATA, AND ALFORD & ASSOCIATES.

FLEX (tm) is a trademark of Technical Systems Consultants, OS-9 (tm) is a trademark of Microware Systems Corporations, MDOS (tm) and EXORciser (tm) are trademarks of Motorola Incorporated.

An SS-50C all CMOS 256K STATIC RAM board will be available SOON!

Write for details & pricing.

WORSTEAD LABORATORIES, NORTH WALSHAM, NORFOLK, ENGLAND. NR28 9SA.
TEL: (0692) 405189
TLX: 97360 SHARET G

statement "hides" the ACCT field and allows an eject to occur between ACCT values. Since the PAGE is set to six, this aligns each set of data to a six line label.

Both label producing methods result in the output of six line "blocks" where the first few lines are filled and the remainder are blank. If desired, the output may be directed to a .OUT file by use of the FLEX "O" command, for additional processing. Here, a possibility would be to print multi-column labels using the Westchester ABS LISTM utility. Another is to free up the system while labels are printing by use of the FLEX print spooling utility.

The mailing list application is potentially part of a larger database. For this reason, a reference such as ACCT should always be included in the file definition as a "link" to other files. This permits name and address information to reside in a single file, yet be accessible to other files via a "key" field. This, of course reduces file space required, allows more data to be stored on each disk and simplifies file updating and maintenance.

Rick Michelhaugh
1104 23rd St.
Portsmouth, Ohio 45662
614 354-4006

'68' Micro Journal
5900 Cassandra Smith Rd
Hixson, Tenn 37343

Mr. Williams,

Over the last few months I have had a rash of problems with my Percom SBC/9. I decided to trash the SBC/9 and replace it with a CPU card from Data Systems '68'. I was pleasantly surprised at the quality of the board. It makes the SBC/9 look awful *cheaply* built. The sol er masking is nice. The switch selected RAM/ROM for $F000 aand $F800 is great! The advertisement did not brag enough about the product. I found the ocumentation very good even though it said 'Preliminary Documentation'. I did find it lacking in the troubleshooting information, although I didn't have any problems, a troubleshooting guide might be helpful in the future. I also have purchased a dual serial interface card. It was also of very good quality and good documentation was enclosed.

Even though Data Systems '68' has since jacked up their prices, I would still recommend their products highly. I plan on buying more of their products in the future.

I would like to thank Dr. Matt Scudiere for his help in setting up my system. My system cosists of:
Data Systems '68' CPU Board, Serial Board
Digital Research 64K Memory Board,
Decwriter LA30 Terminal,
Percom Colorama-50,
SWTPC DC-3 5 1/4 Disk Controller,
Shugart SA400 Disk Drive,
Robertson Electronics
Calendar/Clock/Timer/Parallel Port
16 Ch. A/D Converter Homebuilt from article by John Jordan.

I'm thinking about putting my Colorama-50 on page 1 with 16K of memory. I would like to write some Graphics routines that don't use up any of my 56K of memory. Does anyone else out there own a Colorama-50? I think that this is a darn good board but I never hear anything about it. Is there anyone interested in working with me on this little project? If everyone owning a Colorama-50 would contact me, I would be willing to act as a clearing house for software for the Colorama-50.

Sincerely,

Rick Michelhaugh

Rick Michelhaugh

XMPR...PROM ADAPTOR...REVIEW

ACORN - XMPR 2764 PROM ADAPTOR

Way back in the 'beginning', about 7 or 8 years ago, when SWTPC was the only major Standard S50 Bus manufacturer going (Incidentally they are **THE OLDEST ACTIVE MICROCOMPUTER MANUFACTURER IN THE WORLD!**) there was a new device introduced, the EPROM 'The Electrical Programmable Read Only Memory'. They came in different packages, were hard to program (different voltages on different pins, etc.), sorta undependable(!) and cost a lot for the small amount of storage they allowed. Times and EPROMs have changed!

From 256 bytes or less (which was a lot in those days), to 256K bytes in sample quanities now, the versatile EPROM has surly come a long way. In the process; one of the very first EPROM 'burners' was one sold by SWTPC, in kit form, for less than $50. It originally was designed to program the, then new and current, 2516 and 5v 2716 EPROMs. It came, for the 6800, with a rather nice software package that made the process simple, easy and quick. Many still consider this one of the better EPROM burners available, however, not too many still program the older 2K byte EPROMs. Also SWTPC later (after the 6809 was adopted) came out with software to burn the newer 2532 4K EPROMS, using the same MPR EPROM programmer, but we have never seen anything to update this unit for the latest 2764, 8K EPROMs (Intel type). That is until we received the ACORN Computer Systems XMPR kit. ACORN also claims that the XMPR will work with **any** 2716 PROMburner.

The ACORN XMPR is a 'add-on' kit consisting of a PC board, and all parts except sockets, that converts the 2716 burner to a 2764 burner. Also included is a photographic overlay for the newer configuration.

Additional instructions are included to cover the programming of 2532 types of EPROMs as well. Thus you will be able to handle both the 4K and 8K EPROMs with this one adaptation. Also there are instructions for those having the Radio Shack Color Computer that allows reading of the 68764 EPROM used in the CoCo.

Soft are

Because of the advantages of a 'buffer' type software package, ACORN has available, for $10 on 5" FLEX™ diskette a program quite similar to the SWTPC 6800 EPROM software, but for the 6809. It is furnished in source so that the users may modify his/her port and monitor addresses and assemble. This package is highly recommended to those ordering the XMPR.

Additional information may be secured by calling or writing:

ACORN Computer Systems
11931 W. Bluemound Road
Wauwatosa, WI 53226
(414) 257-0300

PRICE - XMPR $19.95
Software - $10.00

HELP

Help!!! I need a service manual or a schematic diagram and strapping information for PERTEC's 8" Model FD511 disk drives. I will copy and return someones if they will lend. THANKS, Bill Taebel, 348 N. Constitution Dr., Tucson Az. 85748. 602-886-0917.

CLASSIFIED ADVERTISING

Gemini 10 80 column Dot Matrix Printer w/Serial interface $415; Comet 80 Dot Matrix Printer $240; ProWriter 80 column Dot Matrix Printer $440, All three new - still in original boxs.
Two CAL-COMP 8" Disk Drives, $390.00.
Call Tom (615)842-4601.

•••

SWTPC 6809 running FLEX. Card only or complete system. $350 including DC-2 less drives. Call for info.
Dave Pils 203-528-8088.

•••

** QUALITY SOFTWARE NEEDED **

Standard S50 Bus and Color Computer

For the past few months we at the South East Media Division of Computer Publishing, Inc. (CPI), the parent company of 68 MICRO JOURNAL, have debated expanding into the software distribution business. Many other magazines have been doing so for years. Presently there are many fine examples of software that has been developed by **YOU** our readers, that will never see the 'light of day' unless someone, with enough exposure and willingness to continually advertise, runs with the ball.

Software is the 'backbone' for the **real** utilization of any computer, ours are no exceptions! Realizing that there will be some conflicts, with other advertisiers, this has been no simple decision. However, since day one the foremost concern of 68 MICRO JOURNAL has been it's **readers!** Therefore, South East Media Division will accept, for appraisal, software that runs on 6809 systems, games, utility or applications programs.

In the past there has been too much software offered that was not quite ready, nearly, but not quite. We will strive to eliminate that element. But right up front we tell you only that we will do our very best, nothing more. Also we will strive to keep cost to a bare minimum, while securing for the author a fair return, in royalty payments, promptly paid.

Of course we will expect, no — demand, that the author keep the product free of errors (bugs), and maintain it on a prompt and business like basis. Also we shall require that authors be willing to furnish 'source' for those programs that justify, by price and utility, inclusion of same. The lack of source code, properly commented, is a continual complaint we hear. Not all programs will be sold with source, but where necessary, we will insist that it be included.

In some instances the program may be small or short and not justify itself as a 'single' sale product. In this event it will be combined with other like programs, and offered as a package. In that event the royalties will be split between the various authors.

If you have software that you feel will qualify under this program please contact the proper person as shown below.

Color Computer	Standard S50 Bus
Tom Williams	Don Williams
Bob Nay	Bob Nay

Remember, if your software has **any** problems or 'funnies' — GET IT STRAIGHT BEFORE YOU CONTACT US!!!!! Also get your source code in proper shape and well commented. There is too much 99% code already drifting around.

South East Media, POB 794, Hixson, TN 37343 — (615) 842-4601
A Division of CPI

Software

For Ordering Call TOLL FREE 1-800-338-6800

FLEX™ OS-9™ Color Computer

Computer Systems Consultants:

AVAILABLE NOW!!!

The Powerful Super Sleuth Disassembler for the Data-Comp, Hogg, and Spectral Color Comp. FLEX Systems:

Color FLEX Object Code only $50.00
Color FLEX w/ Source Code $99.00
Normal FLEX w/ Source Code $99.00
UNIFLEX w/ Source Code $100.00
OS-9 w/ Source Code $101.00

ALL Comp Sys Cons Programs run on the Color FLEX Systems

Great Plains Computer Co.

AVAILABLE NOW!!!

STYLOGRAPH 2.0 for the Data-Comp and Hogg Color FLEX Systems. Full screen display and editing (i.e., what you see is what you get); supports proportional printers. Operates with the 51 x 24 screens.

SPECIAL COLOR FLEX STYLO $195.00
FLEX STYLO $295.00
UNIFLEX STYLO $395.00

Fast SPELLING CHECKER; allows directly changing the Text File, adding words to the dictionary, etc. 75,000 words in less than 400 sectors (easily fits 5¼" disks with other normal commands on it also).

FLEX, Color FLEX, OS-9 $125.00
UNIFLEX $175.00

MAIL MERGE — greatly extends the power and flexibility of STYLOGRAPH. Allows Multiple Text files to be printed out as one large document. Provides for merging information into the Text File during printing (such as different names and addresses), etc.

FLEX, Color FLEX, OS-9 $145.00
UNIFLEX $195.00

INFOMAG Data Base Management System — An XBASIC-Based, Menu Driven, DBMS with "Built-In" Audit Tracking, Extremely Powerful Report & Format Capabilities, etc. This "Time Proven" DBMS will become the "Work Horse" of your Software "Stable."

FLEX and Color FLEX $295.00
UNIFLEX $395.00

Also in Stock; **Accts Rec., Accts Payable & Gen Ledger** — A full blown Accounting Package that can be used together, or as separate packages; provides the IRS required Audit Tracking. (XBASIC, based on the "Osborne Business Programs.")

FLEX and Color FLEx $295.00/PROG.
UNIFLEX $395.00/PROG.

Also in stock; **PRMAC**; a Relocatable, Recursive Macro-Assembler and Linking Loader for the 6809. An enhancement of Ed Smith Software Works original system. Use either Motorola Format or Special Ed Smith Format. Supports Recursive Macros, Conditional Assembly, etc. All I/O thru normal FLEX System Calls. Can provide a full Cross Reference Listing. Includes a Small Line-oriented Editor as part of the Assembler. Greatly improved operating manual.

FLEX & Color FLEX 6809 PRMAC w/Link & Editor $150.00
SOURCE $ 50.00

Call for information on any requirement—
We have over 200 FLEX, Color FLEX, UNIFLEX, OS-9 and Color Computer Programs available; Games, Programming Languages, Business Software, etc., etc., etc. ??? More Info; just watch these ads!!!, write or call South East Media.

Depend On South East Media

Make Your 'BACK-UP' On South East Media

FREE DISKETTE With Each $50 Purchase

For Ordering Call TOLL FREE 1-800-338-6800

SOUTH EAST MEDIA
5900 Cassandra Smith Rd., Hixson, TN 37343
(615) 842-4601

*FLEX is a trademark of Technical Systems Consultants
*OS9 is a trademark of Microware

O-F

Finally the barrier has been removed from OS9™ to FLEX™ formatted disk!! Now you can **READ** and **WRITE** to a FLEX™ diskette, 5 or 8 inch, with **O-F**.

O-F is a new and unique program, written in BASIC09™ that performs the following functions, and comes complete with source.

1. **REFORMAT:** This module formats a disk that can be read by both OS9™ and FLEX™. Eight or five inch selectable.
2. **FLEX.BAS:** This program does the actual read or write function to the special O-F disk. Also it has the disk format and **DIR** (OS9) commands. All selectable from a user-friendly menu. All selections are interactive and complete including all necessary prompts to the operator.
3. **DIR:** This module (menu selected) allows the disk directory to be printed to the screen, while in BASIC09.

FLEX users can read, write and use the special disk as any other FLEX disk, provided the FLEX directory is not allowed to continue beyond track zero (too many files).

$79.95

WINCHESTER BACKUP UTILITIES

The following utilities allow the backup of any size disk system to **any size diskette.**

By simply inserting diskettes as requested by COPY. MULT, a large disk system (Winchester, etc) may be downloaded to your present floppy disk system, any size. No need to fiddle with directory deletions or any of the other tedious operations that must be done using a normal copy routine.

COPYMULT-CMD understands normal "copy" syntax and always keeps up with files already copied by maintaining directories for both host and receiving disk system, thus eliminating hours of tedious keyboard entries and other time consuming cleanup chores.

BACKUP-CMD is a special program that downloads "random" type files, **any size.**

RESTORE-CMD a special program to restructure copied "random" files for copying, or recopying back to the host system.

FREELINK-CMD a "bonus" utility that "relinks" the free chain of a floppy or hard disk thereby eliminating fragmentation.

**Completely documented source files included. **ALL 4 Programs 99.50 on 8" diskette

CHESS 6809

Requires FLEX™ and **Now Runs On Any Type Terminal**

Features:

- Two display boards.
- Change skill level.
- Swap sides.
- Point scoring system.
- Four levels of play.
- Stop 'Mate in 1-2-3-4' moves.
- Make move and swap sides.
- Play white or black.

$79.95 Specify 5" or 8" disk

This is one of the strongest CHESS programs running on any microcomputer, estimated USCF Rating 1600+.

DIET-TRAC Forecaster

DIET-TRAC Forecaster is a program that plans a diet in terms of either calories and percentage of carbohydrates, proteins and fats (C P G %) or grams of Carbohydrate, Protein and Fat food exchanges of each of the six basic food groups (vegetable, bread, meat, skim milk, fruit and fat) for a specific individual.

Sex, Age, Height, Present Weight, Frame Size, Activity Level and Basal Metabolic Rate for normal individuals are taken into account. Ideal weight and sustaining calories for any weight of the above individual are calculated. When a weight goal is given (either gain or loss), and a calorie plan is agreed upon between the computer and the individual, the number of days to reach the weight goal is projected. The starting and ending rate of weight loss is calculated, and a daily calendar with each day's weight for a 30-day period is printed.

FLEX VERSION — $59.95 UniFLEX VERSION — $89.95

A COLOR COMPUTER TERMINAL DRIVER

TERM is a new and long needed terminal driver for those color computer users who have developed stinging, red and watering eyeballs! TERM allows you to switch from the CoCo keyboard and TV monitor to a real CRT video terminal. Think what that means! No more 32 or even 51 characters per line, that are so blurred that you must guess as to what some characters actually are. No more squinting or cussing. Sounds great? Well TERM gets you away from all that. TERM is called from the CoCo as any other program. You then type the 'RETURN' key on the CRT video terminal keyboard and TERM configures the CRT video terminal to the proper baud rate and you are free of squinting and guessing! To return to the CoCo keyboard (God forbid), you simply type in the command EXTERM from the CRT video terminal keyboard, and you are back where you started from, squints and all.

TERM functions from Data-Comp FLEX, directly and requires only a standard CRT video terminal (any el cheapo will do) or a deluxe terminal, either works just fine. The terminal is connected to the serial port of the CoCo by a standard cable and connector. TERM does NOT function in Radio Shack mode (must be FLEX).

If you want the luxury of 80 characters by 24 lines, or more, depending on the CRT video terminal used, then **TERM** is a must!

ONLY $19.95 FREE with purchase of F-Mate

SOUTH EAST MEDIA 5900 Cassandra Smith Rd., Hixson, TN 37343 (615) 842-4601

SPELLB "Computer Dictionary"

No more "Let your fingers do the walking through the Dictionary" while you are inputting Text with your favorite Editor or Word Processor. **SPELLB**, written by Dan Farnsworth of PALM BEACH SOFTWARE, is more than "another Spelling Checker"; it allows you to "look up a word" from within your Editor or Word Processor so that you KNOW it is right WHEN YOU TYPE IT IN (if your Editor supports a method of passing a Command to FLEX without exiting the Program) with the **SPH.CMD** Utility (which operates in the FLEX Utility Space). Yes, it ALSO allows you to check and update the Text after you are finished; along with allowing you to ADD WORDS to the Dictionary, "Flag" questionable words in the Text for evaluation later, "View" a word in context before changing or ignoring, etc. **SPELLB** first checks a "Common Word Dictionary", then the normal Dictionary, then "Personal Word List", and finally, any "Special Word List" you may have specified. **SPELLB** also allows the use of Small Disk Storage systems.

FLEX and Color FLEX $129.95

"JUST" Text Formatter

JUST, a "Text Formatter" developed by Ron Anderson, provides numerous features which make it a valuable addition to any FLEX Users Software Library. JUST is designed for formatting Text Output for Dot Matrix Printers and provides many unique features:

Output the "Formatted" Text to the Display for format analysis and change.

Output the "Formatted" Text to a Text File for use with the supplied **FPRINT.CMD** for producing multiple copies of the Text on the Printer INCLUDING IMBEDDED PRINTER COMMANDS (this Utility useful at other times also, and worth the price of the program by itself).

"User Configurable" for adapting to other Printers (comes set up for Epson MX-80 with Graftrax); provides for up to ten (10) imbedded "Printer Control Commands", such as Italics on and off, Boldface on and off, etc.

Automatic compensation for a "Double Width" printed line.

Includes the normal line width, margin, indent, paragraph, space, vertical skip lines, page length, page numbering, centering, fill, justification, etc.

Use with ANY Editor.

Supplied with "Structured Source" (Windrush PL/9); easy to see the flow of the program.

FLEX and Color FLEX $49.95

PROGRAMMERS (both NEW and OLD-TIMERS) NOTE —

The Power and Versatility of the 6809 CPU has stretched the capabilities of the "Old Standard" Programming Practices and Procedures, and opened the door for the "Innovators" of the Industry to develop **NEW** and **BETTER** Software Development Tools. **WINDRUSH MICRO SYSTEMS** has stepped forward with three (3) new Programs that will find a home on **EVERY** Programmers work bench.

MACE — XMACE By Graham Trott

MACE is a combined Editor/Assembler designed to allow the Programmer to Enter, Edit, and Assemble Programs with a minimum of effort. **MACE** is designed primarily for the EASY development of small to medium sized Assembly Language programs, but larger programs can be developed using the "Spool" capabilities. The Editor (a simplified Line Editor streamlined for this package) "codes" each Op-Code, providing minimal memory requirements. **MACE** is very "forgiving", which, when combined with the "Interactive" operation, makes this an EXCELLENT package for the Beginning Programmer!

FLEX and Color FLEX — $98.00

PL/9 — By Graham Trott

PL/9 is an Editor/Compiler/Debugger all combined into ONE PACKAGE, which was devised specifically to allow the Assembly Language Programmer the "Best of All Worlds". It allows the Programmer to use "Structured Programming Techniques" while working at the Assembly Language level in a totally **INTERACTIVE** Program Development Cycle (just like working with BASIC; enter some code, try it, edit the code, try it again, etc.). The Single Pass Compiler supports up to 127 Character Symbols; Variable Types; Pointers; Control structures built around the 'Procedure' System, IF..THEN..ELSE, BEGIN..END, WHILE, REPEAT..UNTIL structures, etc., along with Stack, A-, B-, and D-Register manipulation etc. The Editor/Assembler are similar to the **MACE** Program. The Trace/Debugger is oriented towards the **PL/9** Source Program and provides Single Stepping, Breakpointing, running a specified Line Number Range, etc. All in all, this provides an excellent Software Development Tool for utilizing the power of the 6809.

FLEX and Color FLEX — $198.00

C — C-By James McCosh

Finally, for the "Big Time Operator", or for the beginner who can "see the future" and wants to become a part of it, here is the EXCELLENT **WINDRUSH MICRO SYSTEMS** "C Compiler". This is one of **THE** C Compilers for the FLEX Operating System. It can be used with normal Assemblers for most Programming, or with the TSC Relocating Assembler/Linking Loader for those "full blown" System Packages.

FLEX and Color FLEX — $295.00

SOUTH EAST MEDIA 5900 Cassandra Smith Rd., Hixson, TN 37343 (615) 842-4601

FOR ORDERING CALL TOLL FREE 1-800-338-6800

SOUTH EAST MEDIA Software

5900 Cassandra Smith Rd., Hixson, TN 37343
(615) 842-4601

FLEX™ OS-9™ Color Computer

Lucidata

AVAILABLE NOW!!!

PASCAL Language and P-Code Compiler, Version 3. Designed especially for Micro Computer Systems; Run-time System checks available resources for each task, allowing operation on even minimal computer systems. Based on the proposed ISO Standard. Allows linkage to Assembler Code for maximum flexibility.

FLEX and Color FLEX 5" $190.00
FLEX 8" $205.00

PASCAL UTILITIES — Require LUCIDATA Pascal Ver. 3. **XREF** — will produce a Cross Reference Listing of any text; aimed specificlly towards Pascal Source. **INCLUDE** — allows the inclusion of other Files in a Source Text; has unlimited nesting capabilities. Also allows Binary File inclusions. **PROFILER** — produces an Indented, Numbered, "Structogram" of a Pascal Source Text File. Allows viewing the overall structure of large programs, and provides clues as to the integrity of the program. Supplied as Source Code; requires compilation.

FLEX and Color FLEX — Each program $25.00

COPYCAT — Allows reading TSC Mini-FLEX, SSB DOS68, and Digital Research CP/M Disks while operating under FLEX 1.0, FLEX 2.0, or FLEX 9.0 with 6800 or 6809 Systems. COPYCAT will not perform Miracles, but, between the program and the manual, you stand a good chance of accomplishing a transfer. Includes Utilities to List Directories, Copy Files, and convert Text Files when required. Also includes a Utility for investigating Physical Compatibility problems. Programs supplied in Modular Source Code to make it easier to solve unusual problems.

FLEX and Color FLEX 5" $50.00
FLEX 8" $65.00

Westchester Applied Business Systems

AVAILABLE NOW!!!

XDMS Data Management System. Possibly one of the most powerful DMS's available. This machine language program is small enought to operate on a single sided 5" disk, yet provides the speed of M.L. and power limited only by the user's imagination. Supports Sequential, Hierarchical, and Random Access File Structures, and has Virtual Memory capabilities for those Giant Data Bases. Easy-to-use English Language Command Structure.

FLEX and Color FLEX $179.95

For the "**Strictly Color Computer**"Users

AVAILABLE NOW!!!

Hoyt Sterns Electronics—

Intrigued by **Forth**???? Here is a Forth package tailored to the color computer! This package is supplied on Tape, with instructions for transferring it to disk if you wish. Written primarily in machine language, it's speed is unparalelled. A full Semigraphic-8 Editor is provided, along with "goodies" like Graphics and Sound Commands, Printer Commands, Auto-Repeat and Control Keys, etc. If you are interested in Learning Forth, a Trace Feature is provided which is invaluable. If you are a FORTH Pro, this package provides CPU carry Flag accessibility, Fast Task Multiplexing; Clean Interrupt Handling, etc. (Or; you won't "out grow" the Basic capabilities of the Implementation). Combine this package with Leo Brodie's EXCELLENT Book "Starting FORTH", and you will be a FORTH Expert before you know it (and have a lot of fun doing it!).

$58.95

Introl

AVAILABLE NOW!!!

6809 "C" Compiler; generates extremely efficient object code. Output "benchmarks" close to 10MHz 68000 in 8 Bit Operations; 1.5 times faster than a 4 MHz Z80 when using a 2MHz 6809 System (Re. p 43, '68' Micro Journal, May '83).

FLEX, Color FLEX, OS-9 $375.00
UNIFLEX $425.00
One Year Maintenance. $100.00

For Ordering Call TOLL FREE 1-800-338-6800

MasterCard VISA

™FLEX is a trademark of Technical Systems Consultants
™OS9 is a trademark of Microware

SOUTH EAST MEDIA
5900 Cassandra Smith Rd., Hixson, TN 37343
(615) 842-4601

*** RLFORD'S GOOD NEWS ***

SALE SALE SALE SALE SALE SALE

PRE-INVENTORY CLEARANCE SALE!!

Its inventory time around here. To make things easier on ourselves, we want to sell as much of our current inventory as we can. So until stock is depleted, we are offering some FANTASTIC DEALS on selected items. Remember, these prices are ONLY good until our current stock is gone, so call or write to place your order today!

VS-1 SPEECH SYNTHESIZER.....$120

This is the most popular unlimited speech synthesizer available for the SS-50 I/O bus! All units are completely assembled and tested, LESS the SC-01A synthesizer I.C. If you buy this unit while on sale, we will include a copy of our SC-1 text translator program at no additional charge. This gives you a $170 value. You save $50!! Be sure to specify 6800 FLEX , 6809 FLEX , OS-9 , DOS68 or DOS69 operating system and disk size when ordering.

SP-1 SPEECH SYNTHESIZER.....$110

Our popular unlimited speech synthesizer for the Radio Shack Color Computer is also available on this sale. All units are completely assembled and tested, LESS the SC-01A synthesizer I.C. Buy your SP-1 on this sale and we will INCLUDE our SC-1/COLOR translator program which lets your CoCo speak English using nothing more than PRINT statements! Also, we will throw in our COLOR-TREK+ program. This is our COLOR-TREK program adapted for the SP-1. English messages are continually spoken during game play! The action is non-stop. This makes a $170+ value. You save 50+! Software is available on CASSETTE ONLY.

SC-01A SYNTHESIZER IC'S.....$ 55

These chips are the ones used in our synthesizer boards, and are prime, new parts directly from Votrax. We have a limited number of this most popular speech synthesizer I.C. in stock for this sale. Suggested retail price is $85. Discount houses are selling them for $70. You save $15 on the discount house price. We include copies of the Votrax spec sheets with these parts.

SCREDITOR-III.....$150

All of our customers who have used both of the other major word processing packages available for 68xx systems tell us that SCREDITOR III is the best program that they have ever seen (in truth, ONE customer liked one of the others better, but you can't please EVERYONE!). As of July 15th, SCREDITOR III will cost $175. If you have been considering a word processor, or just need a better editor than what you are using now, this is the time to buy SCREDITOR III! For more information about this fantastic program which includes over 150 pages of documentation (and another 240 pages coming soon!), see our ads in the past several issues of Micro-Journal. Be sure to specify 6800 FLEX , 6809 FLEX , OS-9 , DOS68 or DOS69 operating systems and disk size when ordering.

ORDERING INFORMATION...

All sales are FINAL. No returns or refunds, NO open account orders and NO DEALER DISCOUNTS on VS-1, SP-1 or SC-01A purchases for the duration of this sale. Normal dealer discounts and open account terms will apply to SCREDITOR III purchases.

All overseas orders MUST include $10 for shipping. All North American orders under $100 MUST include $5 for shipping. ALL checks MUST be drawn in U.S. dollars on accounts in U.S. banks. No foreign banks or currencies will be accepted under ANY conditions. All Virginia orders MUST include 4% state sales tax. COD orders in the U.S. ONLY, and must be shipped by UPS. All overseas orders shipped by AIR PARCEL POST. We gladly accept VISA and MASTERCARD orders.

Please allow two to three weeks for order processing. Orders paid by check may take longer. Stocks are limited. Any orders received after stocks are exausted will be returned unprocessed.

ALFORD AND ASSOCIATES
PO BOX 6683
RICHMOND VA 23230
PH 804-320-6722

FLEX is a trademark of Technical Systems Consultants, OS-9 is a trademark of Microware Systems Corporation.

The Lord is our primary associate. Too often with the pressures of business today, time is not taken to acknowledge His presence, and too often we fail, due to our human weaknesses, to show His influence in our lives. We thank each of our customers for their patience and prayers for us when we do not reflect Christ in our business activities.

TEN MOST-ASKED QUESTIONS about DYNACALC™

THE ELECTRONIC SPREAD-SHEET FOR 6809 COMPUTERS

1. **What is an electronic spread-sheet, anyway?**
Business people use spread-sheets to organize columns and rows of figures. DYNACALC simulates the operation of a spread-sheet without the mess of paper and pencil. Of course, corrections and changes are a snap. Changing any entered value causes the whole spread-sheet to be re-calculated based on the new constants. This means that you can play, 'what if?' to your heart's content.

2. **Is DYNACALC just for accountants, then?**
Not at all. DYNACALC can be used for just about any type of job. Not only numbers, but alphanumeric messages can be handled. Engineers and other technical users will love DYNACALC's sixteen-digit math and built-in scientific functions. You can build worksheets as large as 256 columns or 256 rows. There's even a built-in sort command, so you can use DYNACALC to manage small data bases — up to 256 records.

3. **What will DYNACALC do for ME?**
That's a good question. Basically the answer is that DYNACALC will let your computer do just about anything you can imagine. Ask your friends who have VisiCalc™, or a similar program, just how useful an electronic spread-sheet program can be for all types of household, business, engineering, and scientific applications. Typical uses include financial planning and budgeting, sales records, bills of material, depreciation schedules, student grade records, job costing, income tax preparation, checkbook balancing, parts inventories, and payroll. But there is no limit to what YOU can do with DYNACALC.

4. **Do I have to learn computer programming?**
NO! DYNACALC is designed to be used by non-programmers, but even a Ph.D. in Computer Science can understand it. Even experienced programmers can get jobs done many times faster with DYNACALC, compared to conventional programming. Built-in HELP messages are provided for quick reference to operating instructions.

5. **Do I have to modify my system to use DYNACALC?**
Nope. DYNACALC uses any standard 6809 configuration, so you don't have to spend money on another CPU board or waste time learning another operating system.

6. **Will DYNACALC read my existing data files?**
You bet! DYNACALC has a beautifully simple method of reading and writing data files, so you can communicate both ways with other programs on your system, such as the Text Editor, Text Processor, Sort/Merge, STYLOGRAPH™ word processor, RMS™ data base system, or other programs written in BASIC, C, PASCAL, FORTRAN, and so on.

7. **How fast is DYNACALC?**
Very. Except for a few seldom-used commands, DYNACALC is memory-resident, so there is little disk I/O to slow things down. The whole data array (worksheet) is in memory, so access to any point is instantaneous. DYNACALC is 100% 6809 machine code for blistering speed.

8. **Is there a version of DYNACALC for MY system?**
Probably. You need a 6809 computer (32k minimum) with FLEX™, UniFLEX™, or OS-9™ operating system. You also need a decent crt terminal, one with at least 80 characters per line, and direct cursor addressing. If your terminal isn't smart enough for DYNACALC, you probably need a new one anyway. The UniFLEX and OS-9 versions of DYNACALC allow you to mix different brands of terminal on the same system. There's also a special version of DYNACALC for Color Computers equipped with FLEX (Frank Hogg or Data-Comp versions).

9. **How much does DYNACALC cost?**
The FLEX versions are just $200 per copy; UniFLEX version $395; OS-9 version (works with LEVEL ONE or LEVEL TWO) $250. Orders outside North America add $7 per copy for postage. We encourage dealers to handle DYNACALC, since it's a product that sells instantly upon demonstration. Call or write on your company letterhead for more information.

10. **Where do I order DYNACALC?**
See your local DYNACALC dealer, or order directly from CSC at the address below. We accept telephone orders from 10 am to 6 pm, Monday through Friday. Call us at 314-576-5020. Your VISA or MasterCard is welcome. Please specify diskette size for FLEX or OS-9 versions. Software serial number is required for the UniFLEX version.

Order your DYNACALC today!

Foreign Dealers:

Australia & Southeast Asia: order from Paris Radio Electronics, 161 Bunnerong Road (PO Box 380) Kingsford, 2032 NSW Australia. Telephone: 02-344-9111.

United Kingdom: order from Compusense, Ltd., PO Box 169, London N13 4HT. Telephone: 01-882-0681.

Scandinavia: order from Swedish Electronics hk AB, Murargatan 23-25, Uppsala S-754 37 Sweden. Telephone: 18-25-30-00.

Computer Systems Center
13461 Olive Blvd.
Chesterfield, MO 63017
(314) 576-5020

UniFLEX software prices include maintenance for the first year.

DYNACALC is a trademark of Computer Systems Center

VisiCalc is a trademark of VisiCorp.
STYLOGRAPH is a trademark of Great Plains Computer Co.
RMS is a trademark of Washington Computer Services.
FLEX and UniFLEX are trademarks of TSC.
OS-9 is a trademark of Microware and Motorola.

OMEGASOFT INDUSTRIAL STRENGTH PASCAL

OMEGASOFT PASCAL

If you're looking for a language to write real-time process control software, look no further. With the rising cost of labor, it is becoming more critical that a high level language be used. Find out why over 500 companies have switched to OmegaSoft Pascal for their demanding applications.

WHY PASCAL?

Pascal was designed to teach students how to write structured programs that are easy to read and maintain. In the past decade it has also proved to provide the same advantages in industrial applications.

EXTENSIONS

OmegaSoft has taken the Pascal framework and expanded the basic data types, operators, functions, and memory allocation to fit the needs of real-time systems. These additions fit in the same structure as Pascal and enhance its usefulness without impairing the excellent readability, ease of maintenance, and structured design.

The byte data type allows you to directly address bytes in memory or I/O devices. The common arithmetic operations can be used for bytes along with shift left, shift right, "and", "or", "eor", and complement operators. These operators are also available for integer and hex (2 byte unsigned) numbers.

Longintegers are four byte signed numbers useful for extended range arithmetic commonly needed for machine control. Functions have been added to allow conversion between the various data types. Dynamic length strings allow complex text manipulation and allow effective interactive I/O.

Variables can be placed either on the data stack (default), at an absolute address in memory (for I/O), in base page, relative to the program counter (for constant tables), or defined in another module.

FEATURES

The compiler generates assembly language for assembly and linking to run on the target system. Since a true relocating assembler and linking loader is used, only those runtime modules required are automatically linked in, providing a much smaller object module than other compilers.

Large Pascal programs can be split up into conveniently sized modules to speed the development process. Procedures, functions, and variables can be referenced between Pascal modules and assembly language modules by using Pascal directives.

Full source code is included for the runtime library, the debugger, and other support utilities.

ISO COMPATIBILITY

OmegaSoft Pascal has been tested using the pascal Validation Suite. The Suite is a collection of over 400 Pascal programs designed to test the quality of Pascal Compilers and their runtime systems for compliance with the ISO (International Standards Organization) Pascal standard. OmegaSoft is the only supplier of 6809 native Pascal compilers that publishes this report in its instruction manual.

DEBUGGER

The compiler package includes an interactive, symbolic debugger. The debugger allows setting of breakpoints, displaying and changing variables, and tracing statements. The debugger allows very fast turnaround for programs to be run on the host system.

TARGET SYSTEMS

The target system may be any 6809 system. No specific I/O devices are required. The output code is re-entrant and rom-able, perfect for single-board systems up to large development systems. There are no charges for use of the output of the compiler or the object of the runtime library in your products. 68000 target coming in 3rd Qtr. 1983.

HOST SYSTEM

The host system must be 6809 based and have at least 48K of ram (56K recommended) and run one of the following operations systems: MDOS, XDOS, OS-9, or FLEX. Priced from $425. 68000 host coming in 4th Qtr. 1983.

SUPPORT PRODUCTS

The OmegaSoft Relocatable Assembler and Linking Loader is designed to support the Pascal Compiler Package and can also be used for general assembly language program development. Priced from $125.

OmegaSoft's Screen Editor supports smart terminals and comes complete with the Pascal source. Priced from $90.

Faster floating point execution can be obtained by use of the Arithmetic Processor Unit option which uses the AMD9511 APU chip. Priced from $90.

For complex real-time applications, the Multi-tasking Kernel provides task scheduling, inter-task communications, and resource interlocking. The Kernel is a runtime library that is accessible as Pascal functions and procedures (with full source included). Priced from $175.

Dealer and OEM inquiries invited. OmegaSoft products are also available from distributors in Australia and Western Europe, call or write for more information.

OMEGASOFT
P.O. Box 842
Camarillo, CA 93010
(805) 987-6426

FEATURES THE POWERFUL, THIRD GENERATION, MOTOROLA 6809 PROCESSOR!

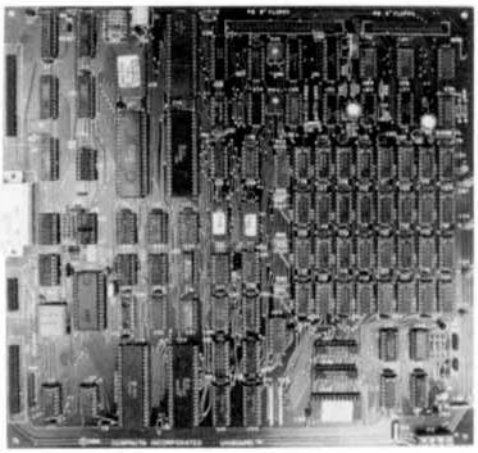
THE 6809 "UNIBOARD"™

SINGLE BOARD COMPUTER KIT

New!

PERFECT FOR COLLEGES, OEM'S, INDUSTRIAL AND SCIENTIFIC USES!

64K RAM! DOUBLE DENSITY FLOPPY DISK CONTROLLER!



BLANK PC BOARD
$99.95
WITH PAL'S, AND TWO EPROMS.

FOR 5-1/4 OR 8 INCH SOURCE DISKETTE ADD $10.

$399.00
COMPLETE KIT! FULLY SOCKETED.

ALL OPTIONS ARE STANDARD. NO EXTRAS TO BUY!

THE COMPACTA UNIBOARD™: Through special arrangement with COMPACTA INC., we are proud to have been selected the exclusive U.S. Mfg. of their new 6809 UNIBOARD™ COMPUTER KIT. Many software professionals feel that the 6809 features probably the most powerful instruction set available today on ANY 8 bit micro. Now, at last, all of that immense computing power is available at a truly unbelievably low price.

FEATURES:

- ★ 64K RAM using 4116 RAMS.
- ★ 6809E Motorola CPU.
- ★ Double Density Floppy Disk Controller for either 5-1/4 or 8 inch drives. Uses WD1793.
- ★ On board 80 x 24 video for a low cost console. Uses 2716 Char. Gen. Programmable Formats. Uses 6845 CRT Controller.
- ★ ASCII keyboard parallel input interface. (6522)
- ★ Serial I/O (6551) for RS232C or 20 MA loop.
- ★ Centronics compatible parallel printer interface. (6522)
- ★ Buss expansion interface with DMA channel. (6844)
- ★ Dual timer for real time clock application.
- ★ Powerful on board system monitor (2732). Features commands such as Go To, Alter, Fill, Move, Display, or Test Memory. Also Read and Write Sectors. Boot Normal, Unknown, and General Flex™.

YOUR CHOICE OF POPULAR DISK OPERATING SYSTEMS:

FLEX™ from TSC	$149
OS9™ from Microware	$199

Specify 5-1/4 or 8 Inch

PC BOARD IS DOUBLE SIDED, PLATED THRU SOLDER MASKED, 11 x 11-1/2 IN.

ALL SALES ARE MADE SUBJECT TO THE TERMS OF OUR 90 DAY LIMITED WARRANTY. A FREE COPY IS AVAILABLE UPON REQUEST.

Digital Research Computers
(OF TEXAS)
P.O. BOX 461565 • GARLAND, TEXAS 75046 • (214) 271-3538

TERMS: Shipments will be made approximately 3 to 6 weeks after we receive your order. VISA, MC, cash accepted. Add $4.00 shipping. USA AND CANADA ONLY

64K SS-50 STATIC RAM

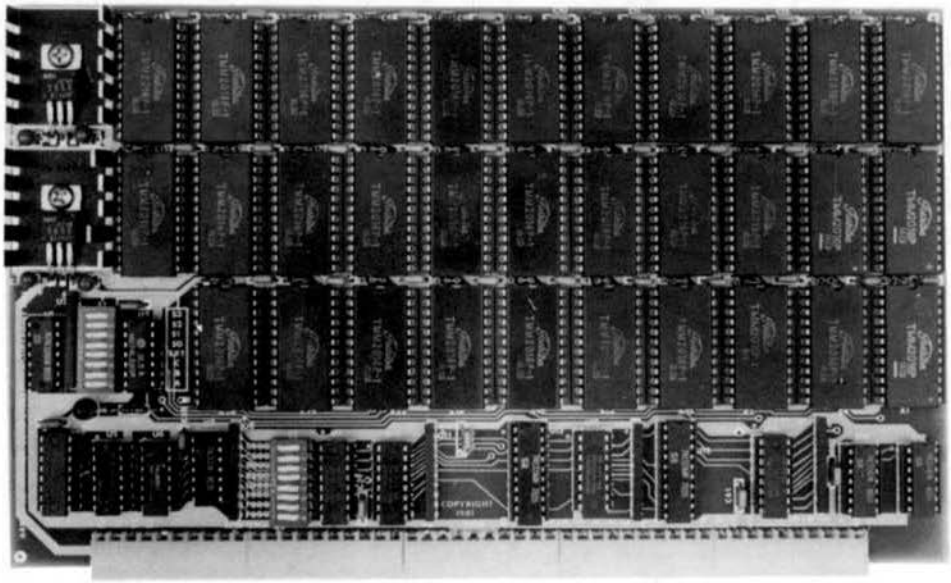
$179.00 (48K KIT)

NEW!

NEW!

LOW POWER!

RAM OR EPROM!



BLANK PC BOARD WITH DOCUMENTATION $52

SUPPORT ICs + CAPS -	$18.00
FULL SOCKET SET -	$15.00

ASSEMBLED AND TESTED **ADD $50**

FEATURES:

- ★ Uses new 2K x 8 (TMM 2016 or HM 6116) RAMs.
- ★ Fully supports Extended Addressing.
- ★ 64K draws only approximately 500 MA.
- ★ 200 NS RAMs are standard. (TOSHIBA makes TMM 2016s as fast as 100 NS. FOR YOUR HIGH SPEED APPLICATIONS.)
- ★ Board is configured as 3-16K blocks and 8-2K blocks (within any 64K block) for maximum flexibility.
- ★ 2716 EPROMs may be installed anywhere on Board.
- ★ Top 16K may be disabled in 2K blocks to avoid any I/O conflicts.
- ★ One Board supports both RAM and EPROM.
- ★ RAM supports 2MHZ operation at no extra charge!
- ★ Board may be partially populated in 16K increments.

56K	$219
64K	$249

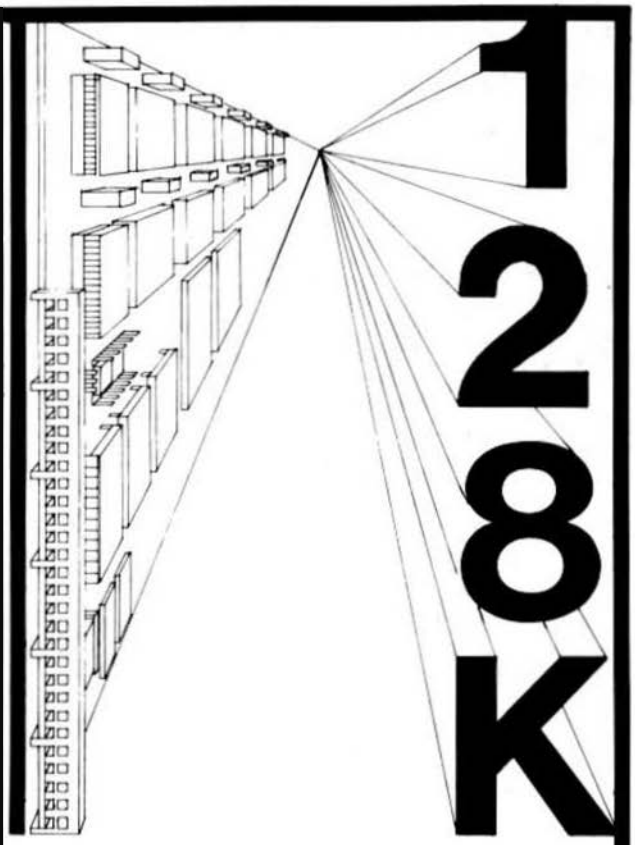
CLOSE OUT SPECIAL
WE HAVE DROPPED OUR 32K SS-50 STATIC RAM BOARD WHICH USED 2114 LOW POWER RAMS. WE WILL SELL THE REMAINING STOCK OF BLANK PCB'S WITH DATA FOR $17.50 EA. THESE FORMERLY SOLD FOR $50.

16K STATIC RAMS?

The new 2K x 8, 24 PIN, static RAMs are the next generation of high density, high speed, low power, RAMs. Pioneered by such companies as HITACHI and TOSHIBA, and soon to be second sourced by most major U.S. manufacturers, these ultra low power parts, feature 2716 compatible pin out. Thus fully interchangeable ROM/RAM boards are at last a reality, and you get BLINDING speed and LOW power thrown in for virtually nothing.

Digital Research Computers
(OF TEXAS)
P.O. BOX 401565 • GARLAND, TEXAS 75040 • (214) 271-3538

TERMS: Add $2.00 postage. We pay balance. Order under $15 add 75¢ handling. No C.O.D. We accept Visa and MasterCharge. Tex. Res. add 5% Tax. Foreign orders (except Canada) add 20% P & H. Orders over $50, add 85¢ for insurance.



Print Buffer

- Holds over 50 average-size pages
- On board firmware with many software features
- RS-232-C serial or centronics type parallel versions available
- MC6809 on board microprocessor
- Plugs directly into SS-30 and SS-30C computers (SWTPc, GIMIX, SSB and others)
- Tested, complete with owner's manual

VCM-128P: Parallel Version	$459.00
VCM-128S: Serial Version	$489.00
Cable Assembly (Serial or Parallel)	$ 19.95

Be Sure To Ask About Our 8K Buffer/Interfaces

Please add $5.00 for USA orders, $15.00 for overseas shipping/handling.

MC/VISA
Dealer Inquiries Welcome

ChiraTech Scientific Instruments

204 N. Link Lane, Alpha 9
Fort Collins, Colorado 80524
(303) 484-1913

ARCADE 50

POWERFUL COLOR GRAPHICS
Uses the new TMS9918A Video Display processor. High resolution 256 x 192 pixel display with 15 colors. 16K Bytes of onboard RAM does not reduce user memory. 32 graphic images can be individually moved with simple X-Y commands for smooth animation. External Video input allows subtitle. NTSC composite video output.

SOUND EFFECTS AND MUSIC
- Three AY3-8910 Programmable Sound Generators
- Nine simultaneous voices
- Three independent noise sources
- Onboard stereo amplifier drives two 8 ohm speakers

ADDITIONAL I/O CAPABILITIES
- Eight analog inputs with 8 bit resolution
- Supports four joysticks with pushbutton switches
- Eight bit parallel I/O port
- Entire unit maps into 256 bytes of memory

FBASIC

TERMINUS DESIGN INC, in conjunction with Microware Systems Corporation, is proud to announce FBASIC an enhancement of Microware's 6800/BASIC. Their fast compiled BASIC has been adapted for 6809 users with added video and sound features for ARCADE 50 users. FBASIC is a true compiler that produces optimized machine language modules which are ROMable and require no Run-Time package. FBASIC requires less memory overhead and runs hundreds of times faster than BASIC interpreters. It supports standard BASIC instruction including String functions. Disk I/O and fast integer arithmetic with multiple-precision capability. Graphics verbs and functions fully support the Arcade 50.

ARCADE 50 assembled and tested	$325.00
Video and Audio connector set	15.00
4 Joystick connector set	15.00
2 Radio Shack Joysticks	24.00
Gold Molex connectors	12.00
A/BASIC for 6800	110.00
FBASIC for 6809	110.00
FBASIC (with ARCADE 50)	75.00
ARCADE 50 RGB	375.00
LABVIDEO (Motorola EXORbug)	375.00
NEW MV09 6809 Processor Board	225.00
256K Dynamic Memory Board	795.00
256K Dynamic Memory Board (w/64K)	395.00
64K Dynamic Memory Board	295.00

TERMINUS DESIGN INC
16 SCARBROUGH ROAD
ELLENWOOD, GA 30049
(404) 474-4866

68 MICRO JOURNAL PROGRAMS on DISK

Disk #1: FILESORT, MINICAT, MINICOPY, MINIFMS, **LIFETIME, **POETRY, **FOODLIST, **DIET.
Disk #2: DISKEDIT w/ inst. & fixes, PRIME, *PRMOD, **SNOOPY, **FOOTBALL, **HEXPAWN, **LIFETIME.
Disk #3: CBUG09, SEC1, SEC2, FIND, TABLE2, INTEXT, DISK-EXP, *DISKSAVE.
Disk #4: MAILING PROGRAM, *FINDDAT, *CHANGE, *TESTDISK.
Disk #5: *DISKFIX 1, *DISKFIX 2, **LETTER, **LOVESIGN, **BLACKJAK, **BOWLING.
Disk #6: **PURCHASE ORDER, INDEX (Disk file indx).
Disk #7: Linking Loader & RLOAD, Harkness
Disk #8: CRTSET, Lanpher (May '82)
Disk #9: DATECOPY, DISKFIX9 (Aug '82)

NOTE: All are as published or received by 68 Micro Journal, some have fixes and patches.

This is a reader service only! No warranty is offered or implied, they are as received and are for reader convenience ONLY. Also 6800 and 6809 programs are mixed, as each is fairly simple (mostly) to convert to the other.

PRICE: 8" Disk $19.95 - 5" Disk $17.95

68 MICRO JOURNAL
POB 794
Hixson, TN 37343
615-842-4600

* indicates 6800, ** indicates BASIC SWTPC or TSC - 6809 no indicator.

MASTER CARD - VISA accepted - Foreign add sufficient postage surface or air!!

SS-50 SOFTWARE

FROM THE WORLD'S LARGEST SUPPLIER OF SOFTWARE FOR FLEX AND OS-9!!!

A/BASIC COMPILER - generates pure, fast efficient 6809 machine code from easy to write BASIC source programs. For 6809 OS-9 or FLEX - **$150.00**

DYNASTAR - Screen Editor - A powerful menu-driven screen editor equally suited to the tasks of program preparation and document processing. For 6809 OS-9 or FLEX - **$149.95**

DYNAFORM - Text Formatter - Provides all the standard features such as pagination, headers and footers with page numbers, single space, double space, bold face, double strike and more. Purchase with DYNASTAR for complete word processing system. For 6809 OS-9 or FLEX - **$149.95**

DYNASOFT PASCAL - A fast integer only P-Code compiler ideal for ROMable applications. Powerful enough for the DYNASTAR word processor. Written for 6809 OS-9 - **$69.95** or FLEX - **$59.95** (Run-time source is additional $30.00)

DYNA "C" - A very fast middle "C" from the people who brought you Dynasoft Pascal.
FLEX - **$99.95** OS-9 - **$124.95**

HELP - A data retrieve utility designed to save you time digging through manuals looking for info about computer language commands and statements. Written in assembler for 6809 FLEX - **$29.95**

AUTOTASK - Consists of a set of memory resident commands that are aids in the design of user oriented applications programs. Written in assembler for 6809 FLEX - **$79.95**
w/MCOMMAND - a utility for converting disk resident commands to memory resident commands-**$50.00** (Free with purchase of Autotask)

VDISK - Provides a way for FLEX users to take advantage of a large memory array. Permits the user to treat extended memory as an additional disk drive. Written for 6809 FLEX - **$100.00**

READTEST - A must for all writers and instructors, it reads your text and tells you how well it was written. In assembler for 6809 FLEX - **$54.95**

CRASMB - Will cross assemble source code into object code. The source or binary available for the following: 6800, 6801, 6809, 6502, 1802, Z80, and Z8. OS-9 version includes 6809 binary. (Use 6809 system to develop software for 6803 Micro Color Computer) OS-9 and FLEX **$200.00**. CPM's additional.

OSM - A fast and versatile macro assembler with ability to define macros with substantial parameters, conditional assembly directives and ability to change value of a label or symbol. Create OS-9 binary files in FLEX and vice versa! For OS-9 or FLEX - **$125.00**

ED/ASM - ED has all the features of TSC's EDITOR with the addition of screen type editing, MACRO capability, and a math package. ASM is compatible with TSC's ASSEMBLER, but with more powerful MACROS and conditionals than TSC's. For 6809 FLEX - **$69.95**

DYNASPELL - The most versatile 68XX spelling checker available. Fast and easy to use. In assembler for 6809 OS-9 or FLEX - **$199.00**

PL-9 -A co-resident EDITOR COMPILER/DEBUGGER written by Graham Trott. A single pass compiler that produces position independent machine code output. For 6809 FLEX - **$198.00** (Exclusive US distributors).

MACE - A co-resident EDITOR ASSEMBLER written by Graham Trott. Includes XMACE, a co-resident 6800/1/3 EDITOR/CROSS ASSEMBLER. For 6809 FLEX - **$98.00**

FHL EXTENDED USE UTILITIES - A set of 12 utilities that add the final touch to your utilities for FLEX. Written in assembler for 6809 FLEX - **$49.95**

TOOLKIT #1 - Add EDIT capabilities to TSC's BASIC, along with DECOMPIL and cross reference. For 6809 FLEX - **$49.95**

TOOLKIT #2 - A package of utilites and programs developed to extend the capabilities of the FLEX operating system. Written in assembler for 6809 FLEX - **$49.95**

FHL FRANK HOGG LABORATORY

THE REGENCY TOWER • SUITE 215 • 770 JAMES ST. • SYRACUSE, NY 13203
PHONE (315)474-7856 • TELEX 646740

'68' MICRO JOURNAL ADVERTISERS INDEX

This index is provided as a reader service. The publisher does not assume any liability for omissions or errors.

TO REALIZE THE FULL POWER & PERFORMANCE OF THE 6809, LOOK TO GIMIX.

GIMIX OFFERS YOU A VARIETY OF SS50 BUS COMPONENTS AND SYSTEMS.

OS-9 GMX III

The GMX 6809 CPU III and OS-9 GMX III. A Multi-user, Multi-tasking package for the ultimate in System Performance plus protection of the system and other users from crashes caused by errors in individual users programs.

#01 (CPU & Software) ..... NEW! **$1695.01**

INTELLIGENT I/O PROCESSOR BOARDS increase system throughput by reducing interrupts to the host, buffering data transfers, and data preprocessing. Prices include on board firmware. Requires system drivers.

#11 3 port RS232 Serial (SS30) ..... NEW! **$498.11**
#12 4 port Parallell (SS50) **$388.12**
OS-9 GMXIII drivers . . (included when purchased with GMX III package) . . **$200.00**
OS-9 Level 2 users -- contact GIMIX for system requirements and availability.

192K GMX III #79 SYSTEMS: All include GMX 6809 CPU III and OS-9 GMX III (#01); a #11 3 port intelligent serial I/O & cables; #19 Classy Chassis; 192KB Static RAM, #68 DMA controller, all necessary cables, power regulators, and filler plates. The OS-9 Editor, Assembler, Debugger, BASIC-09, and RUNB are included.

#79 with dual 40 track DSDD drives **$5998.79**
#79 with dual 80 track DSDD drives ..... NEW! **$6298.79**
#79 with #88 8" Dual Drive Disk System **$7598.79**
#79 with #90 19MB Winchester subsystem & one 80 track DSDD drive . . **$8698.79**
UniFLEX for the GMX 6809 CPU III and Intelligent I/O boards is in development.

OS-9 GMX I; OS-9 GMX II; FLEX; and UniFLEX

The **#05** GIMIX 6809 PLUS CPU board **$578.05**
Options: GMX DAT ..... **$35.00** SWTPC DAT **$15.00**
9511A ..... **$312.00** 9512 ..... **$265.00**

#49 64KB GHOST SYSTEM includes: #05 CPU; #19 Classy Chassis; 64KB static RAM; a #43 2 port serial card & cables; #68 DMA Controller; all necessary cables, power regulators, and filler plates; GMXBUG monitor; FLEX; and OS-9 GMX I. You can software select either FLEX or OS-9. The OS-9 Editor, Assembler, Debugger, BASIC-09, and RUNB are also included.

#49 with dual 40 track DSDD drives **$4398.49**
#49 with dual 80 track DSDD drives **$4698.49**
#49 with #88 8" Dual Drive Disk System **$5998.49**
#49 with #90 19MB Winchester subsystems & one 80 track DSDD drive **$7398.49**

#39 128KB SYSTEM includes: #05 CPUwDAT; #19 Classy Chassis; 128KB of static RAM; a # 3 2 port serial card & cables; #68 DMA Controller; all necessary cables, power regulators, and filler plates; GMXBUG monitor; FLEX; and OS-9 GMX II. You can software select either FLEX or OS-9. The OS-9 Editor, Assembler, Debugger, BASIC-09, and RUNB, and GMX-VDISK for FLEX are included.

#39 with dual 40 track DSDD drives **$4998.39**
#39 with dual 80 track DSDD drives **$5298.39**
#39 with #88 8" Dual Drive Disk System **$6598.39**
#39 with #90 19MB Winchester subsystem & one 80 track DSDD drive **$7998.39**
UniFLEX, available at extra cost, requires 8" or Winchester drives. A signed license agreement with TSC is required before shipment.

You can add to any GIMIX system RAM, I/Os and other options, or substitute non-volatile RAM. GIMIX will customize to your needs.

COMING SOON: Contact GIMIX for price and availability on 40MB and 72MB Winchester (5¼") drives, removeable pack Winchesters, 256KB static RAM boards.

All GIMIX systems are guaranteed for 2MHz operation. GIMIX systems include documentation for all boards and software in a GIMIX binder. **ALL DRIVES ARE 100% TESTED AND ALIGNED BY GIMIX.**

ALL BOARDS AND SYSTEMS ARE ASSEMBLED, BURNED-IN, AND TESTED. GOLD-PLATED BUS CONNECTORS ARE USED.

TO ORDER BY MAIL: SEND CHECK OR MONEY ORDER OR USE YOUR VISA OR MASTER CHARGE. Please allow 3 weeks for personal checks to clear. U.S. orders add $5 handling if order is under $200.00. Foreign orders add $10 handling if order is under $200.00. Foreign orders over $200.00 will be shipped via Emery Air Freight COLLECT, and we will charge no handling. All orders must be prepaid in U.S. funds. Please note that foreign checks have been taking about 8 weeks for collection so we would advise wiring money, or checks drawn on a bank account in the U.S. Our bank is the Continental Illinois National Bank of Chicago, 231 S. LaSalle Street, Chicago, IL 60693, account #73-32033. Visa or Master Charge also accepted.

EXPORT MODELS: ADD $30 FOR 50Hz. POWER SUPPLIES.

GIMIX Inc. reserves the right to change pricing, terms, and product specifications at any time without further notice.

ALL PRICES ARE F.O.B. CHICAGO

Choose from GIMIX' wide variety of system components.

The GIMIX CLASSY CHASSIS #19 consists of a heavyweight aluminum cabinet, constant voltage ferro-resonant power supply, and SS50 Mother board with baud rate generator board **$1398.19**

Triple Disk regulator card and cables **$88.22** Baud rate generator card **$98.93**
Missing cycle detector ..... **$38.23** Filler plates ..... **$1 .92**
Back panel connector plates (specify) . **$8.60** 50 Hz. option **$30.00**

MEMORIES (GIMIX uses only Static RAM)

#67 64KB NMOSSTATIC RAM board **$478.67**
#64 64KB CMOS STATIC RAM board w/battery back-up **$568.64**
#34 8K PROM board **$98.34**
#32 16 socket PROM/ROM/RAM board **$238.32**

I/O Boards (see above for Intelligent I/Os)

#41 Single port serial, RS232/20ma. current loop **$88.41**
 # 3 2 port serial, RS232 ..... **$128.43**
#46 8 port serial, RS232 ..... **$318.46**
#42 2 port parallel **$88.42**
#45 8 port parallel **$198.45**
#50 serial, RS232, RS422, RS423 ..... **$244.50**
#52 SSDA serial, RS232, RS 22, RS423 ..... **$254.52**
#54 AOLC serial, RS232, RS 22, RS 23 ..... **$268.54**
Each cable with connectors for back panel mounting (specify board) **$24.95**

DISK CONTROLLERS

#68 DMA (featured in all systems above) **$588.68**
#28 dbl. dens. programmed I/O (5" drives only) **$298.28**
#58 single dens. programmed I/O (5" and/or 8" drives) **$226.58**
#48 same as #58 but for 5" drives only **$198.48**
Cable sets: 8" with Back Panel connector **$29.25**
for two 8" external drives **$44.26**
for two 5" drives **$34.98**

SOFTWARE: GIMIX exclusive versions of OS-9/GMX I, II, III & FLEX are for GIMIX hardware only. All versions of OS-9 require the #68 controller.
When ordered with any controller, FLEX is **$30.00**

GMXBUG PROMs and manual **$98.65**
Boot or Video boot PROM ..... **$30.00** UNIFLEX boot PROM **$50.00**
OS-9 GMX I ..... **$200.00** OS-9 GMX II **$500.00**
Editor ..... **$125.00** Assembler **$125.00**
BASIC-09 ..... **$200.00** RUNB **$100.00**

DISK DRIVES FOR GIMIX SYSTEMS -- complete with cables and power regulators.

5" DSDD 40 track ..... 2 for **$900.00**
5" DSDD 80 track ..... 2 for **$1300.00**
#88 8" Dual 8" DSDD drives, cabinet, power supply, & cables **$2698.88**
Cabinet only ..... **$848.18** 220V 50Hz. Option, add **$30.00**
Filler plate ..... **$14.83** Cable for 2 drives **$44.82**
Cable for 4 drives ..... **$57.84** Cable for cabinet to mainframe **$45.81**

WINCHESTER SUBSYSTEMS: for use only in GIMIX systems with #68 DMA controller.

#90: includes one 19MB drive, interface, and Software **$5498.90**
#91: includes two 19MB drives, interface and Software **$5288.91**
Contact GIMIX for price and availability of other forthcoming subsystems.

OTHER BOARDS

#76 GHOST 80X24 VIDEO BOARD **$398.76**
#65 50 pin Protoboards ..... **$56.66** **#33** 30 pin Protoboards **$38.33**
#03 6800 CPU **$224.03**
#06 6800 CPU with timers ..... **$288.06** Baud rate option, add **$30.00**

#08 RELAY DRIVER (board, bracket, transformer, and 3± relays) **$1128.08**
#86 - #08 (board, bracket, transformer, without relays) **$538.86**
#85 OPTO board **$348.85**
WINDRUSH EPROM PROGRAMMER **$375.00**
3" Binder ..... **12.00** 2" Binder **$9.00**

GIMIX DOES NOT GUARANTEE PERFORMANCE OF ANY GIMIX SYSTEMS, BOARDS OR SOFTWARE WHEN USED WITH OTHER MANUFACTURERS PRODUCT.

DON'T SEE IT??? ASK! OUR BROCHURE HAS MORE COMPLETE DESCRIPTIONS AND SPECS. PHONE OR WRITE TODAY FOR YOUR COPY.

BASIC-09 and OS-9 are trademarks of Microware Systems Corp. and MOTOROLA, Inc. FLEX and UniFLEX are trademarks of Technical Systems Consultants, Inc. GIMIX, GHOST, GMX, CLASSY CHASSIS, are trademarks of GIMIX, Inc.

GIMIX INC. 1337 WEST 37th PLACE • CHICAGO, ILLINOIS 60609
(312) 927-5510 • TWX 910-221-4055

The Original FLEX™ for Color Computers

- Upgrade to 64K
- RS to FLEX, FLEX to RS file transfer ability
- Create your own character set
- Automatic recognition of single or double density and single or doubled sided
- All features available for either single or multiple drive systems
- Settable Disk Drive Seek Rates
- Faster High Resolution Video Display with 5 different formats
- Save RS Basic from RAM to Disk
- Move RS Basic to RAM
- Load and save function on FLEX disk
- 24 Support Commands 12 with Source Text
- External Terminal Program

Languages Available

Pascal, Fortran, RS Basic, RS Assembler, TSC Basic, TSC Assembler, Relocating Assembler, Macro Assembler, Mumps

If you are tired of playing games on your TRS-80C Color Computer, or find that you are handicapped by the limitations of the RS BASIC in trying to write a Program that will allow you to actually USE the Color Computer as a COMPUTER, YOU ARE READY TO MOVE UP TO THE **FLEX9** Operating System. If you want to have REAL PROGRAMMING POWER, using an Extremely Powerful Business BASIC, PASCALs, C Compilers, a full-blown Macro Assembler with a Library capability so you are not continuously reinventing the wheel... YOU ARE READY TO MOVE UP TO THE **FLEX9** Operating System. If you would like to see if YOU REALLY COULD USE A COMPUTER IN YOUR BUSINESS, or begin to make your Computer start PAYING IT'S OWN WAY by doing some Computer Work for the millions of small businesses around you, such as Wordprocessing, Payroll, Accounting, Inventory, etc., then YOU ARE READY TO MOVE UP TO THE **FLEX9** Operating System. How?? **DATA-COMP** has the way!

DATA-COMP's FLEX9 Conversion for the TRS-80C Color Computer was designed for the SERIOUS COMPUTER **USER**, with features like greatly increased Display Screens, WITH Lower Case Letters, so you can put a FULL Menu on ONE Screen, or see SEVERAL Paragraphs at the same time, with features like providing a FULL Keyboard so you have FULL Control of your Computer AND it's Programs NATURALLY, without needing a chart to see what Key Combination will give you what function, with USER ORIENTED functions to make using the Operating System natural, like having the Computer AUTOMATICALLY determine what type of Disk is being used in what type of Disk Drive and working accordingly, rather that you have to specify each and every thing for it, or like having the Computer work with the Printer you have been using all along without you having to tell the new Operating System what is there, etc.

™FLEX is a Trademark of Technical Systems Consultants

DATA-COMP has everything you need to make your TRS-80C Color Computer WORK for YOU: from Parts and Pieces to Full, Ready To Use SYSTEMS. DATA-COMP designs, sells, services, and SUPPORTS Computer SYSTEMS, not just Software. CALL DATA-COMP TODAY to make your Computer WORK FOR YOU!

System Requirements

FLEX9 Special General Version + Editor & Assembler (which normally sell for $50.00 ea.)	$150.00
F-MATE(RS) FLEX9 Conversion Rout. for the RS Disk Controller	
when purchased with Special General FLEX9 Sys	$49.95
when purchased without the General FLEX9 Sys	$59.95
Set of Eight 64K RAM Chips w Mod. Instructions	$59.95
Color Computer with 64K RAM and EXT. BASIC	$399.95

SPECIAL SYSTEM PACKAGES

64K Radio Shack COLOR COMPUTER, Radio Shack COLOR DISK CONTROLLER, a Disk Drive System, Special General Version of FLEX9™, F-MATE(RS)™ and a Box of 10 Double Density Diskettes; a COMPLETE, ready to run SYSTEM on your Color TV Set. $1079.95

DISK DRIVE PACKAGES, etc.

These Packages include the Radio Shack Disk Controller, Disk Drives with Power Supply and Cabinet, and Disk Drive Cable.

PAK #1 — 1 Single Sided, Double Density Sys	$489.95
PAK #2 — 2 Single Sided, Double Density Sys	$749.95
PAK #3 — 1 Double Sided, Double Density Sys	$569.95
PAK #4 — 2 Double Sided, Double Density Sys	$919.95
PAK #5 — 2 Qume Thinline, Double Sided Double Density Sys	$749.95

PARTS AND PIECES

Radio Shack Disk Controller	$179.95
1 Single Sided, Double Density Disk Drive Tandon	$249.95
1 Double Sided, Double Density Disk Drive Qume	$349.95
1 Qume Thinline, Double Sided, Double Density	$279.95
Screen Clean — Clears Up Video Distortion On Your Color Computer	$39.95
Single Drive Cabinet with Power Supply	$89.95
Double Drive Cabinet with Power Supply	$109.95
Single Drive Disk Cable for RS Controller	$24.95
Double Drive Disk Cable for RS Controller	$34.95
Micro Tech Prods., Inc. **LOWER CASE ROM** Adapter	$74.95
Radio Shack BASIC Version 1.1 ROM	$34.95
Radio Shack Extended Basic ROM	$89.95

TERM — AN External Terminal Driver is FREE with purchase of F-Mate

---

LAST OF PRODUCTION!!!

!! LIMITED QUANTITY !!

SWTPC 8212 & 8212(W) Intelligent Terminals

Data-Comp Has Bought the Remaining Supply of SWTPC 8212 CRT Terminals

If You Need an 8212 We Have Them In Stock!

PRINTERS

The Epson MX-80
$425.00
The Epson MX-100
$725.00
MX-70 $355.00 MX-80 FT $575.00

☆

Verbatim Diskettes.

5 Soft Sector Disks

Single Side Single Density	$2.75 ea
Single Side Double Density	$2.75 ea
Double Side Double Density	$4.92 ea
Plastic Storage Box	$2.00 ea

8 Soft Sector Disks

Single Side Single Density	$3.75 ea
Single Side Double Density	$4.10 ea
Double Side Double Density	$4.75 ea
Plastic Library Box	$5.00 ea

Foreign Orders Add 10% Surface 20% Air Mail

MasterCard VISA

DATA-COMP

P.O. Box 794 *HIXSON, TN 37343*
1-615-842-4601

EM-82 Video Terminal

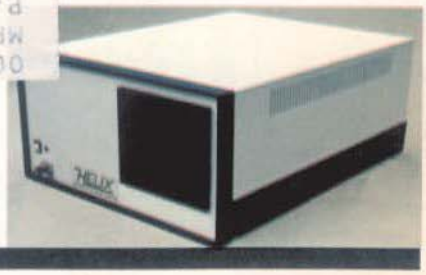
Emulates the 8200 Series from SWTPC

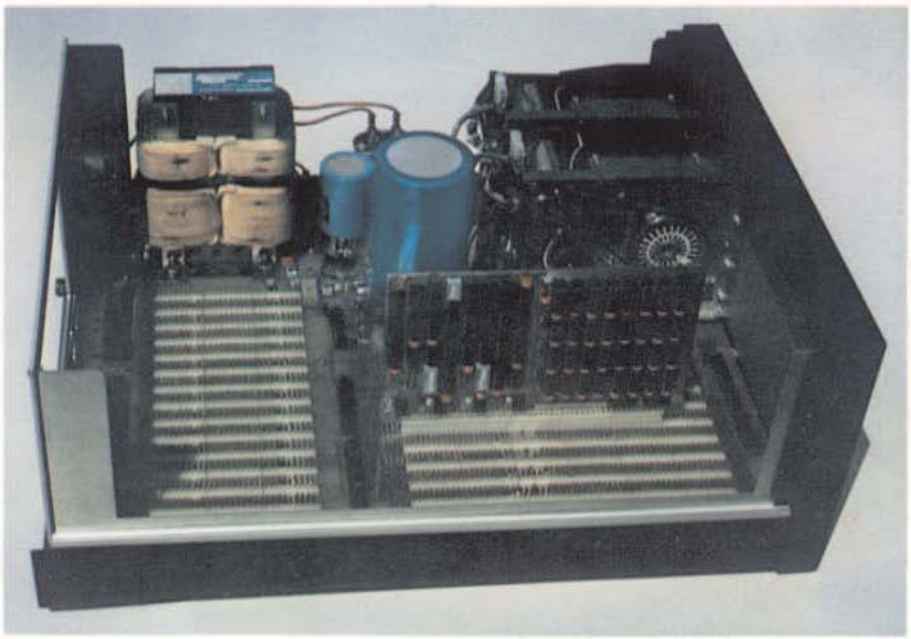
Data-Comp--South East Media & 68 Micro Journal Are Divisions of CPI

68 Micro Journal
5900 Cassandra Smith Rd.
Hixson, TN 37343

Second Class Postage Paid
At Chattanooga, TN
ISSN 0194-5025

HELIX™





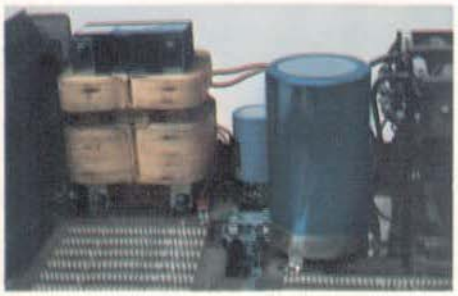
THE MAINFRAME

- Industry Standard Optima™ Cabinet
- Largest Constant Voltage Power Supply in the Industry
- S-64 Bus gives 16 Bit Power and S-50 Bus Compatibility
- 10 Main (S-64) Slots
- 14 I/O (S-30) Slots plus 2 On-board
- On-board Baud Rate Generator to 38.4Kb
- Space and Power for two 5¼" Disk Drives
- Full Address Decoding for I/O Slots
- Two RS-232 Serial and Two parallel Ports On-board
- Single Board Construction for Reliability
- Faraday Shielded Bus Lines give "Text Book Clean" Signals

THE PROCESSORS

6809

- Standard 2 MHz Operation
- Standard DAT Compatible with GIMIX and SWTPC
- Standard 6840 Interval Timer
- Standard 1K Scratchpad RAM
- Standard Clock/Calendar with Battery
- Provision for Programmers Console



The HELIX™ computer system represents the latest advance in S-50 bus computer systems. Relying on the physical nature of S-50 bus connectors to guarantee compatibility, the HELIX adds 14 bus lines (becoming S-64) to allow a 68000 processor to operate with full 16 bit data transfer and 24 bit addressing, while at the same time providing full interchangability with existing S-50 components.

Offered with a selection of processors, memories, and peripheral controllers, a HELIX system can be configured for applications ranging from advanced hobbyist to multiterminal time-sharing.

Designed to offer the utmost in speed, reliability, and utility at a reasonable price, it represents a new standard of quality for those who require a professionally designed computer for professional use.

THE POWER SUPPLY

- Ferro-resonant Transformer for Line Noise and Under-Voltage Protection
- Conservative 25 Amps at 8.5 Volts
- Conservative 5 Amps at ±16 Volts
- Conservative Component Rating for Reliability

THE COMPONENTS

- Fully Socketed
- Gold Plated Bus Connectors
- Only "B" Series 68XX Components Used
- Only Top Grade Logic Circuits Used
- Industrial Grade Components Throughout

THE MEMORIES

DM-64

- Field Proven
- Proprietary Memory Control Logic
- Fully Transparent Refresh
- Tested at 2.5 MHz Operation

DM-512

- 512K Bytes on a Single S-64 Board
- 16 Bit Power and 8 Bit Compatibility
- Runs in Existing S-50 Systems where Physical Space Allows
- Full 24 Bit Addressing
- Fully Transparent Refresh

THE PRICES

Because of the variety of configurations possible, full pricing cannot be given. Representative prices are:

- **64K 6809 HELIX..... $2495**
- **256K 6809 HELIX..... $2895**
- **512K 6809 HELIX..... $3750**

HAZELWOOD COMPUTER SYSTEMS

907 E. Terra, O'Fallon, Missouri 63366 (314) 281-1055

Dealer and OEM Inquiries Invited. We support our Dealers.

Optima is a Trademark of Scientific-Atlanta